PASSIVE LOCALIZATION OF UNDERWATER ACOUSTIC BEACONS

by
Dennis Michael Wojcik

B.S. Electrical Engineering
Marquette University (1986)

Submitted to the
Massachusetts Institute of Technology/
Woods Hole Oceanographic Institution
Joint Program in Oceanography and
Applied Ocean Science and Engineering in
Partial Fulfillment of the Requirements of the Degree of

Master of Science in Oceanographic Engineering

at the
Massachusetts Institute of Technology and the
Woods Hole Oceanographic Institution

September 1993

Signature of Author Dennis Michael Wojcik
Joint Program in Applied Ocean Science and Oceanographic Engineering
Massachusetts Institute of Technology/Woods Hole Oceanographic Institution

Certified by Albert M Bradley
Dr. Albert M. Bradley
Senior Engineer, Woods Hole Oceanographic Institute

Accepted by ______________________
Prof. Arthur N. Baggeroer
Chairman, Joint Committee for Applied Ocean Science and Engineering

PASSIVE LOCALIZATION OF UNDERWATER ACOUSTIC BEACONS

by
Dennis Michael Wojcik

B.S. Electrical Engineering
Marquette University (1986)

Submitted to the Massachusetts Institute of Technology/
Woods Hole Oceanographic Institution Joint Program in
Oceanography and Applied Ocean Science and Engineering in
Partial Fulfillment of the Requirements of the Degree of
Master of Science in Oceanographic Engineering

This thesis examines the use of a single, omnidirectional hydrophone as a receiving sensor to passively localize an acoustic beacon. The localization problem is presented as a constrained, nonlinear parameter estimation problem, and Lagrange multipliers are introduced to solve for the maximum likelihood estimate of the acoustic beacon's position. An iterative algorithm is developed using range difference measurements to solve for the maximum likelihood estimate of a stationary acoustic beacon's position. This algorithm is then extended to include linear, constant velocity motion of the acoustic beacon. Finally, design specifications for a receiver to implement the maximum likelihood estimation algorithms are developed.

To test the maximum likelihood estimate algorithms, Monte Carlo simulations are conducted. Results from six representative scenarios are presented. Test results show that as the number of range differences used increases, or the distance that the observer travels between received beacon signals increases, the accuracy of the estimated position improves. Also, tests show that accuracy of the estimated beacon position is directly related to the accuracy in which the observer's position is measured. To test the receiver's design specifications, a prototype receiver is built using commonly available components. It is then shown that the prototype receiver meets or exceeds the design specifications.

Table of Contents

List of Symbols

C	speed of sound in sea water
e_μ	residual error vector
e_μ^*	final residual error vectors
ε_i	tolerance factor used to stop iterating
$g(w_\mu, \theta)$	structural model of the parameter estimation problem
$G(W, \theta)$	augmented vector of the structural model
G_i	augmented vector of the structural model evaluated at $(\hat{W}^*, \theta^*)$
λ	Lagrange multipliers
Λ	Lagrangian function
m_μ	measurement vector
M	moment matrix of the final estimated residual vectors
n	number of experiment vectors used to estimate parameter vector θ
$\Phi(W)$	objective function
θ	parameter vector describing the acoustic beacon's position and velocity.
θ^*	estimated parameter vector
t_μ	time of receipt of a beacon signal
$t_{s\mu}$	measured synchronized time of receipt
$\hat{t}_{s\mu}$	true synchronized time of receipt
$\tilde{t}_{s\mu}$	measurement errors of the synchronized time of receipt
t_β	time that the acoustic beacon transmits a signal
T_o	time that the acoustic beacon transmits the first signal received
T_p	pulse repetition interval
T_R	acoustic travel time of the signal
T_s	sampling rate of the receiver
$\sigma^2_{t\mu}$	variance of the synchronized time of receipt measurement error
$\sigma^2_{x\mu}$, $\sigma^2_{y\mu}$	variance of the measured ship's position error
V_μ	covariance matrix of the measurement errors
w_μ	measured experiment vector

$\hat{w}_\mu$	true value of the experiment vector
$\tilde{w}_\mu$	measurement errors of the experiment vector
$\hat{w}_\mu^*$	estimated true experiment vector
$\mathbf{W}$	augmented vector of *n* experiment vectors
x_μ, y_μ	measure location of the receiving ship
$\hat{x}_\mu$, $\hat{y}_\mu$	true location of the receiving ship
$\tilde{x}_\mu$, $\tilde{y}_\mu$	measurement errors of the location of the receiving ship
x_T, y_T, z_T	the acoustic beacon' position for the stationary case
x_o,y_o, z_o	the acoustic beacon's initial position for the moving beacon case
x_R, y_R	origin of the local Cartesian coordinate system
x_B, y_B, z_B	acoustic beacon position used to generate synthetic measurement vectors

Chapter I

Introduction and Problem Statement

1.1 Introduction

There are many scenarios in which passive localization of an acoustic beacon is needed. Techniques like Target Motion Analysis and Bearings Only Ranging can passively estimate an acoustic beacon's position by using received bearing information. These techniques rely upon a multiple sensor receiving array to find the relative bearing between the acoustic beacon and the observer. To estimate the position of the

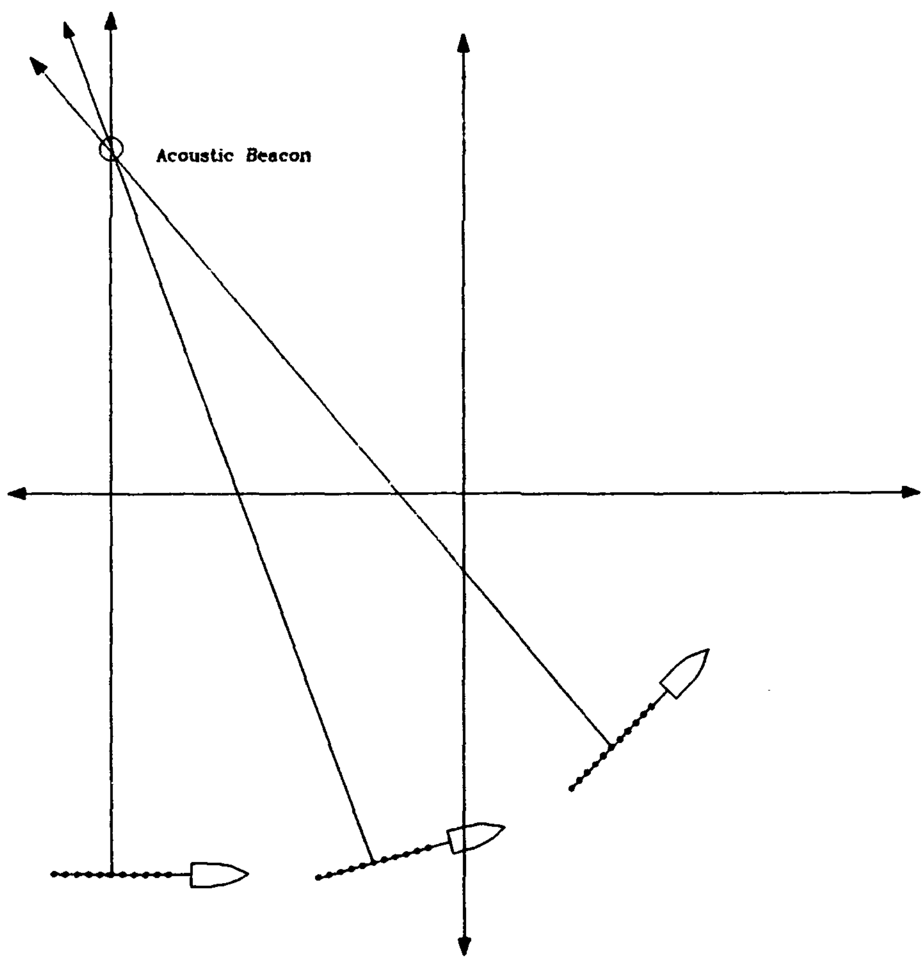


Figure 1-1

acoustic beacon, bearing lines from several different observation points are combined to form a fix. See figure 1-1.

Recent developments in passive target ranging and tracking which use arrays of only a few elements can also be used to passively localize an acoustic beacon [Allen and King 1988] [Friedlander 1987] [Moose 1987]. These methods rely upon the array to estimate the relative bearing between the acoustic beacon and the observer as well. Statistical and test analyses have shown that the performance of the localization techniques mentioned above improves with increased distance between sensor elements in the receiving array [Moose 1987]. This typically leads to the use of long receiving arrays. The use of long arrays can become a problem if the ship used to deploy the array is small, or if budget constraints limit the size of the array. For these cases, the observer can employ a technique that uses a single omnidirectional sensor to form a synthetic aperture array instead of a conventional receiving array to passively localize the acoustic beacon.

Previous techniques to localize acoustic beacons using a single omnidirectional hydrophone instead of an array use a raster scanning paper recording device to produce a mark on recording paper whenever a beacon signal is received. The scan rate of the paper recorder is matched to the pulse repetition interval of the acoustic beacon. This synchronizes the paper recorder, so that a new line is printed for each pulse repetition interval. The observer can note the relative change in travel time between successive beacon signals by comparing the locations of the corresponding marks on the recording paper. If the observer and the acoustic beacon are stationary,

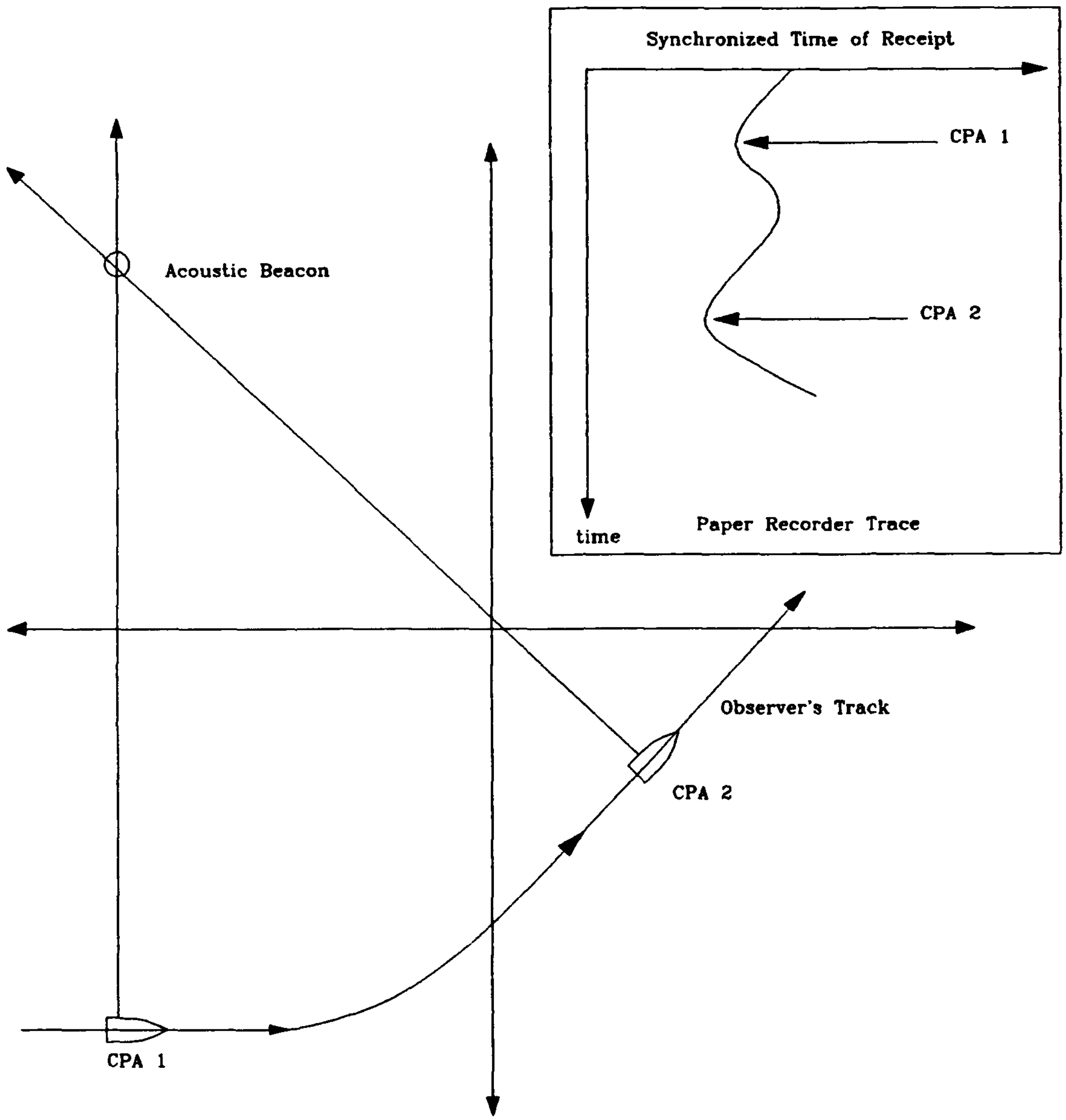


Figure 1-2

the time it takes the acoustic beacon's signal to travel between the beacon and the observer is constant. For this case, the trace made by the paper recorder is a vertical line. As the observer approaches the beacon along a linear path, the range to the beacon, and consequently the time that the beacon signal takes to travel to the

observer, decreases hyperbolically. The corresponding trace made by the paper recorder is also hyperbolic. The observer can use this relation to fix the acoustic beacon's position. To obtain a line of bearing to the acoustic beacon, the observer watches for the closet point of approach (CPA) to the acoustic beacon by noting the point where the trace produced by the paper recorder changes from negative to positive slope. This corresponds to the transverse axis of the hyperbolic trace and is the point where the range changes from decreasing to increasing. At the CPA, the acoustic beacon is located on a line of bearing perpendicular to the direction of the observer's motion. To estimate the location of the acoustic beacon, the observer must obtain lines of bearing from at least two different directions of motion. See figure 1-2. This method of passively locating acoustic beacons relies heavily upon the operator's skill in visually identifying the closest point of approach from the printed output of the paper recorder. Also, the observer is forced to travel in a straight line for a significant period of time until the CPA can be determined. Other factors that limit the usefulness of the above method are that paper recorders tend to be bulky, awkward to move, and require an adequate supply of expensive recording paper.

This thesis examines the use of a single omnidirectional hydrophone in place of a conventional receiving array to efficiently localize an acoustic beacon using a nonlinear parameter estimation technique. The goal is to develop an inexpensive, easily deployed localization receiver that can interface to a personal computer for data analysis and signal processing. As a backup, this receiver must also perform the functions of the older paper recording system, using the personal computer as the

display device.

1.2 Organization

This thesis is organized into six chapters and three appendices:

- Chapter I contains an introduction to the problem of passively localizing acoustic beacons.
- Chapter II contains the development of a technique to localize a stationary acoustic beacon using a constrained Maximum Likelihood Estimator (MLE).
- Chapter III contains an extension of the solution to the stationary acoustic beacon localization problem that allows for constant velocity, linear motion of the acoustic beacon.
- Chapter IV contains results of Monte Carlo simulations of the localization algorithms developed in chapters II and III. The effects of using different effective sensor spacings and different numbers of experimental data points are explored.
- Chapter V contains design criteria for a receiver to implement the algorithms developed in chapters II and III. A test receiver is designed and built based upon these criteria, and experiments are conducted to measure the actual performance of the receiver.
- Chapter VI contains observations and concluding remarks.
- Appendix 1 contains the schematic diagrams of the receiver designed in chapter V.
- Appendix 2 lists the source code for the 68HC11 microcontroller used in the

receiver designed in chapter V.

- Appendix 3 lists the source code for the Macintosh computer used for data analysis and signal processing.

Chapter II

Stationary Acoustic Beacon

2.1 Introduction

When the acoustic beacon is stationary, we are interested in locating the acoustic beacon to within a certain error bound as efficiently as possible. To accomplish this we treat the acoustic beacon's location as an unknown parameter vector, and formulate the localization problem as a constrained multiple dimension parameter estimation problem. To solve the constrained parameter estimation problem, we first develop a structural model that relates the values of a set of available measurements to the unknown parameter vector. We then define a residual error vector as the difference between the true values and the actual values of the measurements. Assuming the measurement errors are normally distributed, independent random variables, we define the likelihood function for a particular set of measurements as the logarithm of the joint probability density function of the measurements errors, viewed as a function of the residual error vectors. Next, we form a set of equality constraints by evaluating the structural model at the true values of the measurements, and the true value of the parameter vector θ. To find the maximum likelihood estimate (MLE) of the unknown parameter vector, we introduce a vector of Lagrange multipliers, and then define a Lagrangian function as the combination of the likelihood function and the sum of the equality constraints, weighted by the Lagrange multipliers. The maximum likelihood estimate is then found by determining the stationary point of the Lagrangian function.

2.2 Problem Geometry

Consider the acoustic beacon to be at a fixed latitude, longitude, and depth. Since we are only concerned with locating the acoustic beacon in a small region of the ocean, we approximate the earth as being flat in the region around the beacon's position and use a Cartesian coordinate system to describe the location of the acoustic beacon. The use of a Cartesian coordinate system does not contribute any significant errors as long as we correct for distortions due to the projection of the acoustic beacon's global coordinates onto the local Cartesian coordinate system. We define a parameter vector θ as:

$$\theta = [\, x_T \;\; y_T \;\; z_T \,]^T \tag{2-1}$$

where x_T represents the east-west position of the acoustic beacon, y_T represents the north-south position of the acoustic beacon, and z_T represents the depth of the acoustic beacon in meters relative to the center of the local Cartesian coordinate system. To correct for the distortions caused by projecting the acoustic beacon's global coordinates onto the local Cartesian coordinate system, we introduce a correction factor that is proportional to the cosine of the latitude. We then relate θ to the acoustic beacon's global position by:

$$\theta = \begin{bmatrix} x_T \\ y_T \\ z_T \end{bmatrix} = \begin{bmatrix} \textit{degrees longitude} \cdot \dfrac{60\ Nm}{1^o} \cdot \dfrac{1853.2\ m}{1\ Nm} \cdot \cos(\textit{degrees latitude}) - x_R \\ \textit{degrees latitude} \cdot \dfrac{60\ Nm}{1^o} \cdot \dfrac{1853.2\ m}{1\ Nm} - y_R \\ -\ \textit{depth} \end{bmatrix} \tag{2-2}$$

where x_R and y_R define an arbitrary reference point on the surface of the ocean which defines the origin of the local Cartesian coordinate system. In equation (2-2) we assume that the acoustic beacon's latitude and longitude are given in degrees.

Similarly, we consider an observer on a receiving ship in the general location of the acoustic beacon. We define a measurement vector m_μ as:

$$m_\mu = [\, x_\mu \quad y_\mu \quad t_\mu \,]^T \tag{2-3}$$

where x_μ and y_μ are the receiving ship's coordinates on the same local Cartesian coordinate system defined for the acoustic beacon, and t_μ is the time that the observer's position x_μ, y_μ was obtained. We use the same relation described in equation (2-2) to convert the receiving ship's global coordinates to local Cartesian coordinates. The measurement vector m_μ is then given by:

$$m_\mu = \begin{bmatrix} x_\mu \\ y_\mu \\ t_\mu \end{bmatrix} = \begin{bmatrix} \textit{degrees longitude} \cdot \dfrac{60\ Nm}{1^o} \cdot \dfrac{1853.2\ m}{1\ Nm} \cdot \cos(\textit{degrees latitude}) - x_R \\ \textit{degrees latitude} \cdot \dfrac{60\ Nm}{1^o} \cdot \dfrac{1853.2\ m}{1\ Nm} - y_R \\ t_\mu \end{bmatrix} \tag{2-4}$$

Again, we assume that the receiving ship's latitude and longitude are given in degrees. Figure 2-1 summarizes the problem geometry.

Now let us assume that the acoustic beacon transmits a signal at a regular interval, and that at time t_μ the observer receives one of the signals from the acoustic beacon. The time t_μ can be related to the time that the signal took to reach the

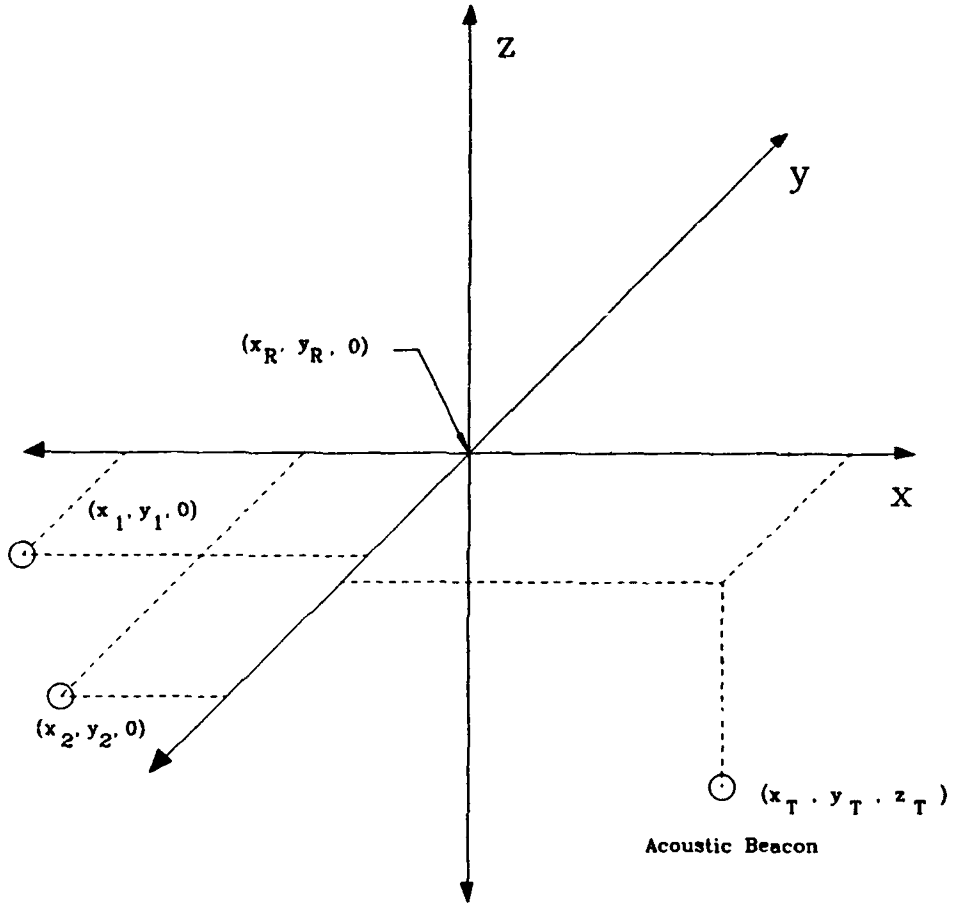


Figure 2-1

observer by:

$$t_{\mu} = T_R + i \cdot T_p + T_o \tag{2-5}$$

Here T_R represents the acoustic travel time of the signal, T_o is the time that the acoustic beacon transmitted the first signal, *i* is the number of pulse repetition intervals between when the first signal was transmitted and when the current signal was transmitted, and T_p represents the pulse repetition interval. The product $i \cdot T_p$ represents the time after T_o that the acoustic beacon transmitted the current received signal. It is

assumed that the observer knows the pulse repetition interval of the acoustic beacon. The observer can therefore remove the dependance upon $i \cdot T_p$ by counting the number of signals received (i), then subtracting $i \cdot T_p$ from equation (2-5). This synchronizes the observer's receiver with the pulse repetition interval of the acoustic beacon. We define the synchronized time of receipt as the time of receipt t_μ corrected to eliminate the pulse repetition interval of the acoustic beacon as:

$$t_{s\mu} \equiv t_\mu - i \cdot T_p = T_R + T_o \tag{2-6}$$

and subsequently redefine the measurement vector m_μ as:

$$m_\mu = [\ x_\mu \quad y_\mu \quad t_{s\mu}\ ]^T \tag{2-7}$$

The difficulty in passively ranging an acoustic beacon arises because the observer does not know T_o, the time the acoustic beacon started to transmit. In active ranging systems, the observer can measure T_R directly by transmitting a signal at a known time T_o, and measuring the time that the corresponding reflected signal from a target is received. In passive ranging systems, the observer does not know when the beacon signal was transmitted and therefore the observer has no time reference T_o. Consequently the observer has no way to directly calculate T_R. In order to passively range a target, the observer must eliminate the dependence on the absolute time reference T_o.

2.3 Structural Model

In order to proceed, we must develop a model which relates the measurement vector m_μ, and the unknown parameter vector θ. We define a function $\mathbf{f}(\mathbf{m}_\mu, \theta)$

which relates the distance between the acoustic beacon's position θ and the receiving ship's position x_μ, y_μ to the synchronized time of receipt $t_{s\mu}$ as:

$$f(m_\mu,\ \theta) \equiv \sqrt{(x_\mu - x_T)^2 + (y_\mu - y_T)^2 + z_T^2} - \tau(C,\ t_{s\mu}) \tag{2-8}$$

where $\tau(C, t_{s\mu})$ is a function that relates the speed of sound in seawater to the measured time of receipt, synchronized to the pulse repetition interval of the acoustic beacon. It is assumed that the receiving ship is on the surface of the ocean, or equivalently z_μ is zero. Using a homogeneous ocean model and a direct path propagation mode for simplicity, we can define $\tau(C, t_{s\mu})$ from equation (2-6) as:

$$\tau(C,\ t_{s\mu}) = C \cdot T_R = C \cdot t_{s\mu} - C \cdot T_o \tag{2-9}$$

Substituting equation (2-9) into equation (2-8) yields:

$$f(m_\mu,\ \theta) = \sqrt{(x_\mu - x_T)^2 + (y_\mu - y_T)^2 + z_T^2} - (C \cdot t_{s\mu} - C \cdot T_o) \tag{2-10}$$

It is apparent from equation (2-10) that we can not eliminate the dependence upon T_o with only one received signal from the acoustic beacon. One way around this problem is to compare the time of receipt and positions from two separate received signals. This is commonly referred to as range difference, or time difference ranging. We define a new function $g(m_1, m_2, \theta)$ as:

$$g(m_1,\ m_2,\ \theta) \equiv f(m_1,\ \theta) - f(m_2,\ \theta) \tag{2-11}$$

Substituting equation (2-10) into equation (2-11) with m_1 for the first received signal and m_2 for the second received signal we have:

$$g(m_1, m_2, \theta) = \sqrt{(x_1 - x_T)^2 + (y_1 - y_T)^2 + z_T^2} + \ldots$$
$$-\sqrt{(x_2 - x_T)^2 + (y_2 - y_T)^2 + z_T^2} + \ldots \qquad \textbf{(2-12)}$$
$$- C \cdot (t_{s1} - t_{s2})$$

The first two terms of the function $\mathbf{g}(\mathbf{m}_1, \mathbf{m}_2, \theta)$ represent the slant range between the acoustic beacon and the receiving ship at the two separate times that a beacon signal is received. By subtracting one range from the other, we are left with the distance that the observer moved radially, relative to the beacon, between the times t_{s1} and t_{s2}. This is known as the range difference between the two points given by m_1 and m_2. The last term of the function $\mathbf{g}(\mathbf{m}_1, \mathbf{m}_2, \theta)$ represent the range difference between points m_1 and m_2 calculated from the time difference. From the last term of $\mathbf{g}(\mathbf{m}_1, \mathbf{m}_2, \theta)$ we see that the dependence upon T_o is eliminated, and we are left with a model that relies on measuring the time of receipt, synchronized to the pulse repetition interval of the acoustic beacon, and the location of the receiving ship at the time of receipt.

To simplify notation, we define an experiment to be the augmentation of two measurement vectors m_1 and m_2, and use the notation:

$$w_\mu = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = [\, x_1 \;\; y_1 \;\; t_{s1} \;\; x_2 \;\; y_2 \;\; t_{s2} \,]^T \qquad \textbf{(2-13)}$$

We then redefine the equation (2-12) as a scaler valued function of the experiment vector w_μ and the acoustic beacon location vector θ:

$$g(w_\mu, \theta) = g(m_1, m_2, \theta) \qquad \textbf{(2-14)}$$

We denote the function $\mathbf{g}(\mathbf{w}_\mu, \theta)$ as the structural model of the parameter estimation problem. Ideally, if the measurements have no errors and $\tau(\mathbf{C}, \mathbf{t}_{s\mu})$ is the correct propagation model, $\mathbf{g}(\mathbf{w}_\mu, \theta)$ equals zero at the true value of the parameter vector θ. The estimation problem is then to determine the value of the parameter vector θ that minimizes the value of the structural model given experiment vectors which contain errors.

2.4 Measurement Errors

The structural model describes the ideal situation - perfect measurements, free from any errors. In reality both the synchronized time of receipt $t_{s\mu}$ and the positions x_μ, y_μ have errors associated with the measurement process. We can define the actual measurements as the sum of the true value and a random error component:

$$w_\mu = \hat{w}_\mu + \tilde{w}_\mu = \begin{bmatrix} \hat{x}_1 + \tilde{x}_1 \\ \hat{y}_1 + \tilde{y}_1 \\ \hat{t}_{s1} + \tilde{t}_{s1} \\ \hat{x}_2 + \tilde{x}_2 \\ \hat{y}_2 + \tilde{y}_2 \\ \hat{t}_{s2} + \tilde{t}_{s2} \end{bmatrix} \tag{2-15}$$

where $\hat{x}_\mu$, $\hat{y}_\mu$, $\hat{t}_{s\mu}$, and $\hat{w}_\mu$ represent the true values of the measurements, and $\tilde{x}_\mu$, $\tilde{y}_\mu$, $\tilde{t}_{s\mu}$, and $\tilde{w}_\mu$ represent the random errors associated with the measurement process.

With the assumption of a homogeneous, direct path propagation mode, and since the time of receipt will be determined by looking at the output of a simple digital peak detector, the errors in measuring $\hat{t}_{s\mu}$ arise mainly from the sampling of the received signal. Because we use a simple peak detector to measure the time of

receipt, we can only estimate the true time of receipt to within one sampling interval - if we sample the received signal every millisecond, then we can only know the time of receipt to within 1 millisecond. This quantization of the time base causes the measurement errors to have a uniform distribution centered around the true value $\hat{t}_{s\mu}$ with a variance given by:

$$\sigma^2_{t_\mu} = \frac{T_s^2}{12} \tag{2-16}$$

where T_S is the sampling interval in seconds. For the sake of computational efficiency and mathematical tractability, we would like to model the measurement errors of $t_{s\mu}$ as zero mean, normally distributed random variables. If we consider that the structural model is not exact (a far more complex function $\tau(C, t_{s\mu})$ is needed to have an exact structural model), and that there will be errors due to factors such as dispersion and multipath interference, we can lump the modeling errors of $\tau(C, t_{s\mu})$ together with the quantization errors in the time $t_{s\mu}$. Although not strictly rigorous, this allows us to simplify the development of the solution to the localization problem by assuming the measurement errors of $t_{s\mu}$ are zero mean, normally distributed random variables, with a variance approximated by $\sigma_{t\mu}^2$. Also, since the measurement of the time of receipt of one signal is not effected by the measurement of any other time of receipt, we can take the errors in measuring $t_{s\mu}$ as independent between t_{s1} and t_{s2}, and also independent between experiments w_μ.

Errors in measuring the ship's position at the time of receipt of the beacon signal are due to the accuracy of the system used to obtain the ship's position. To

obtain the ship's location, a Global Positioning System (GPS) receiver will be used. Other navigation systems such as Loran, Omega, or even Inertial Navigation systems could be used, however the price and accuracy of the GPS system make it the natural choice. Currently, inexpensive commercial C/A code GPS receivers are available with published errors on the order of 25 meters rms in latitude and longitude. The more accurate Differential GPS and P code receivers are not readily available at this time. We will assume that the errors of the GPS positional data are normally distributed, and independent in latitude and longitude. This assumption is unlikely to cause much harm as long as the true probability distribution of the errors have smooth tails [Bard 1974].

With these arguments in mind, and for the sake of computational efficiency and mathematical tractability, we model the measurement errors as zero mean, normally distributed, independent random variables. We take the covariance matrix of the measurement errors as:

$$V_\mu = \begin{bmatrix} \sigma_{x_\mu}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{y_\mu}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{t_\mu}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x_\mu}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{y_\mu}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{t_\mu}^2 \end{bmatrix} \tag{2-17}$$

Where $\sigma_{t\mu}^2$ is the variance given by equation (2-16), σ_x^2 is the variance of the GPS longitude error in meters, and σ_y^2 is the variance of the GPS latitude error in meters.

2.5 Parameter Estimation Solution - Maximum Likelihood Estimator

To solve for the parameter vector θ we would normally use a weighted least square approach. However, because of the nonlinearities in the structural model $\mathbf{g}(\mathbf{w}_\mu, \theta)$, and because all of the measurements have errors, the weighted least squares method does not work well. To demonstrate why the normal weighted least squares method does not work well, we look at the contours of the χ^2 merit function:

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{g(w_\mu, \theta)}{\sigma_{t_\mu}} \right)^2 \tag{2-18}$$

If the structural model is truly exact, and there are no measurement errors in w_μ, $\mathbf{g}(\mathbf{w}_\mu, \theta)$ and χ^2 are equal to zero for the true value of θ. As the value of θ in equation (2-18) moves farther away from the true value of θ, the χ^2 function increases. Figure 2-2 shows the contours of the χ^2 function for the true values of the measurements w_μ, where the acoustic beacon is located at x_T = -1000, y_T = 1000, and z_T = -1000. From Figure 2-2 we see that the contours of the χ^2 merit function that encircle the true position of the acoustic beacon are highly eccentric. This eccentricity leads to large variances in the location estimate along the major axis of the ellipse. With errors in all of the measurements, this in turn produces errors in the parameter estimates that are unacceptable.

To overcome the difficulties in solving the parameter estimation problem with a nonlinear structural model and errors in all of the measurements, we treat the problem as a constrained minimization problem. Our goal is to estimate the true values of the measurements w_μ given a set of *n* equality constraints which we obtain by setting the

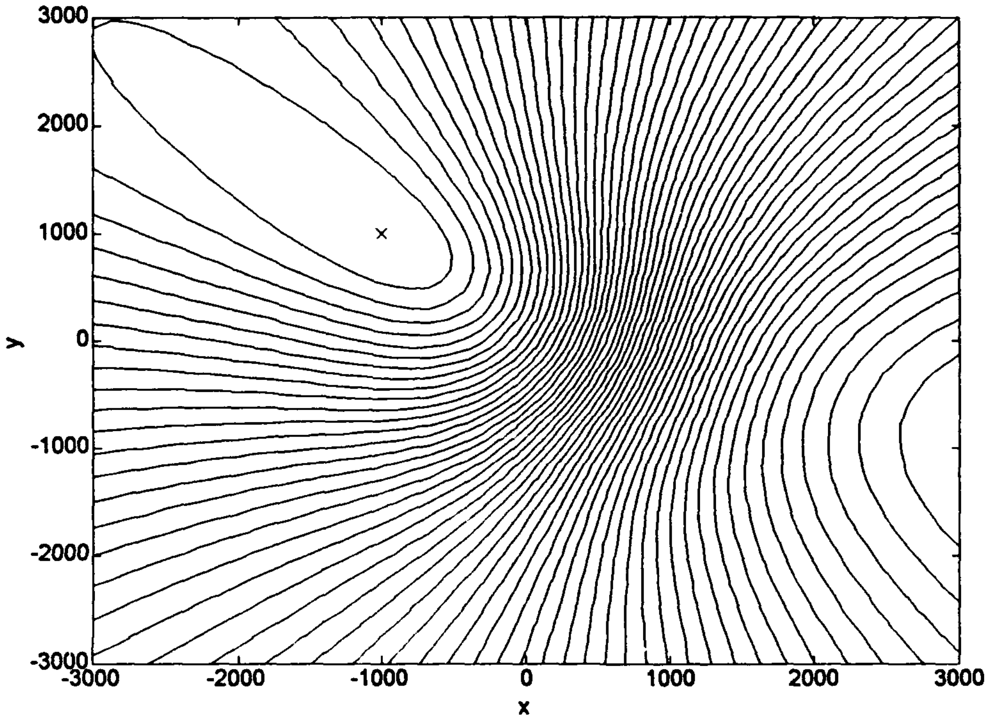


Figure 2-2 Chi Square Contours for the Normal Weighted Least Squares Method

structural model $g(\mathbf{w}_\mu, \theta)$ equal to zero at the estimated true values of the measurements and the estimated parameter vector θ. Because of the measurement errors, we must modify the structural model to indicate the dependence upon the true values of w_μ:

$$
\begin{aligned}
g(\hat{w}_\mu, \theta) = & \sqrt{(\hat{x}_1 - x_T)^2 + (\hat{y}_1 - y_T)^2 + z_T^2} + \ldots \\
& -\sqrt{(\hat{x}_2 - x_T)^2 + (\hat{y}_2 - y_T)^2 + z_T^2} + \ldots \\
& -C \cdot (\hat{t}_{s1} - \hat{t}_{s2})
\end{aligned}
\tag{2-19}
$$

Next we define a residual vector as the difference between the true value and the observed value of the measurements:

$$e_\mu(\hat{w}_\mu) = \hat{w}_\mu - w_\mu \tag{2-20}$$

We also define a composite experiment vector **W** as the augmentation of *n* experiment vectors w_μ:

$$W = \left[\, w_1^T \;\; w_2^T \;\; \dots \;\; w_n^T \,\right]^T \tag{2-21}$$

With the assumption that the measurement errors are normally distributed random variables, independently distributed between experiments, and with the further assumption that the covariance matrix is the same for each experiment, the likelihood function takes the form:

$$\log(L(\hat{W}, V)) = -(n/2)\log(\det V) - \frac{1}{2}\cdot\sum_{\mu=1}^{n} e_\mu^T V^{-1} e_\mu \tag{2-22}$$

where *n* is the number of experiment vectors w_μ used in the estimation problem [Bard 1974]. The maximum likelihood estimate is found by determining the value of θ^*, and the value of w_μ^* for each of the *n* experiments, which minimizes log(L) while satisfying the *n* equality constraints imposed by the structural model. The terms w_μ^* and θ^* represent the maximum likelihood estimate of the true values of w_μ and θ.

To find the maximum likelihood estimate given the *n* equality constraints imposed by the structural model, we introduce an *n*-dimensional vector of Lagrange multipliers λ_μ for each experiment. We then form the Lagrangian function:

$$\Lambda(\hat{W}, \theta, V, \lambda_1, \dots \lambda_n) = \log(L(\hat{W}, V)) + \sum_{\mu=1}^{n} \lambda_\mu \cdot g(\hat{w}_\mu, \theta) \tag{2-23}$$

The solution to the estimation problem will be found at a stationary point of the Lagrangian $\Lambda(W, \theta, V, \lambda_1, \ldots, \lambda_\mu)$ [Bard 1974].

To demonstrate the benefits of using this approach, we look at the contours of the Lagrangian function given by equation (2-23). Using the algorithm developed in section 2.6, figure 2-3 shows the contours of the Lagrangian function obtained by using the same experiment vectors used to generate the contours of the normal weighted least squares method.

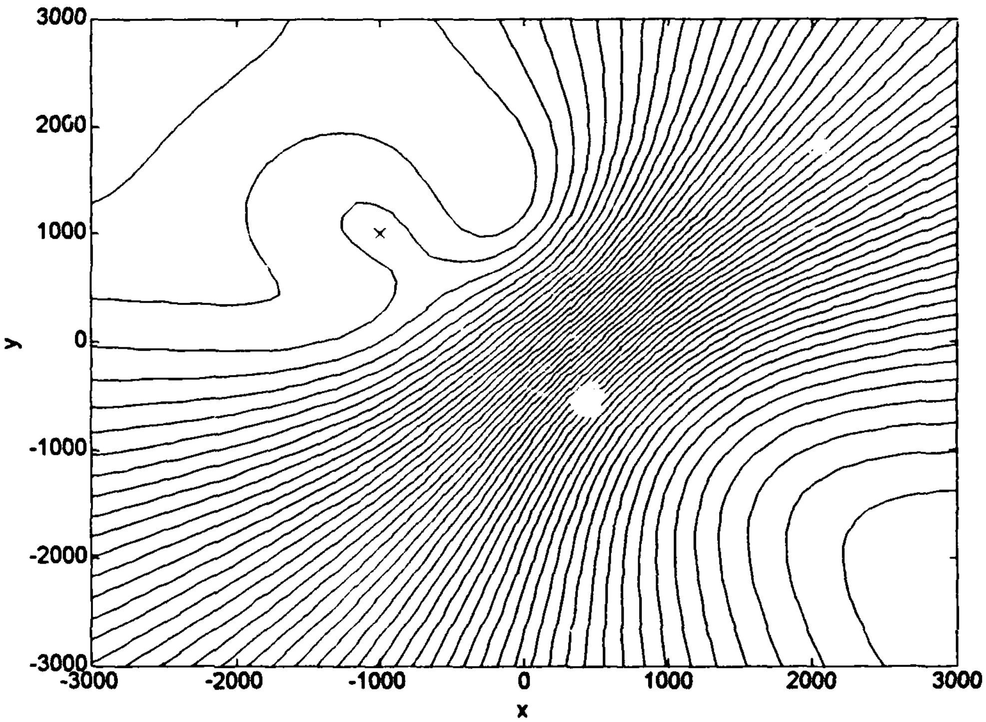


Figure 2-3 Contours of the Lagrangian Function Λ

From figure 2-3 we see that the contours that surround the true position of the acoustic beacon are much less eccentric then the contours of the normal weighted least squares method. Therefore, we do not have the large variances associated with the major axis

of the normal weighted least squares method, and the estimate of the parameter vector θ will be more accurate.

To solve the maximum likelihood estimator problem given the equality constraint of the structural model we use an iterative method suggested by Deming as described by Bard [Bard 1974]. We define the parameter vector θ_i^* as the estimate of the acoustic beacon's location at the *i*th iteration:

$$\theta_i^* = [\, x_{T_i}^* \;\; y_{T_i}^* \;\; z_{T_i}^* \,]^T \tag{2-24}$$

We also define the estimated true experiment vector at the *i*th iteration as:

$$\hat{w}_{\mu_i}^* = [\, \hat{x}_{1_i}^* \;\; \hat{y}_{1}^* \;\; \hat{t}_{s1_i}^* \;\; \hat{x}_{2_i}^* \;\; \hat{y}_{2_i}^* \;\; \hat{t}_{s2_i}^* \,]^T \tag{2-25}$$

and the corresponding composite experiment vector:

$$\hat{W}_i^* = [\, \hat{w}_1^{*T}{}_i \;\; \hat{w}_2^{*T}{}_i \cdots \hat{w}_n^{*T}{}_i \,]^T \tag{2-26}$$

The vector $\hat{W}_i^*$ is the augmentation of the n estimated true experiment vectors $\hat{w}_\mu^*$ at the *i*th iteration. We then define a composite vector valued function $G(\hat{W}_i^*, \theta_i^*)$ where the *j*th component of $G(\hat{W}_i^*, \theta_i^*)$ is the structural model evaluated at $(\hat{w}_\mu^*, \theta_i^*)$:

$$G(\hat{W}_i^*, \theta_i^*) = G_i = [\, g(\hat{w}_{1_i}^*, \theta_i^*) \;\; g(\hat{w}_{2_i}^*, \theta_i^*) \;\; \ldots \;\; g(\hat{w}_{n_i}^*, \theta_i^*) \,]^T \tag{2-27}$$

Next we define an objective function as the weighted least squares sum of the

residuals given by equation (2-‸). At the *i*th iteration the objective function is:

$$\Phi(\hat{W}_i^*) = \frac{1}{2} \cdot \sum_{\mu=1}^{n} (\hat{w}_{\mu_i}^* - w_\mu)^T V_\mu^{-1} (\hat{w}_{\mu_i}^* - w_\mu) \tag{2-28}$$

This is the same form as the last term in equation (2-22). To solve for the Maximum Likelihood Estimate, the objective is to find the $\hat{W}_i^*$ which minimizes log(L), or equivalently, $\Phi(\hat{W}_i^*)$ subject to the constraints $G(\hat{W}_i^*, \theta_i^*) = 0$. We note that now the unknown parameters that we are estimating are the acoustic beacon's location as well as the true values of all the measurements.

To develop a solution, we define the following terms:

$$q_i \equiv \partial\Phi(\hat{W}_i^*)/\partial\hat{W} \tag{2-29}$$

$$H_i \equiv \partial^2\Phi(\hat{W}_i^*)/\partial\hat{W}\partial\hat{W} \tag{2-30}$$

$$A_i \equiv \partial G(\hat{W}_i^*, \theta_i^*)/\partial\hat{W} \tag{2-31}$$

$$B_i \equiv \partial G(\hat{W}_i^*, \theta_i^*)/\partial\theta \tag{2-32}$$

Next we expand the objective function in a Taylor series around the current estimate of **W** and θ. Keeping only up to the second order terms we have:

$$\tilde{\Phi}_i(\delta\hat{W}) = \Phi_i + q_i^T \delta\hat{W} + \frac{1}{2}\delta\hat{W}^T H_i \delta\hat{W} \tag{2-33}$$

Now we expand the structural model in a Taylor series around the current estimate of **W** and θ. Keeping only the first order terms we have:

$$\tilde{G}_i(\delta\hat{W},\ \delta\theta) \equiv G_i + A_i\delta\hat{W} + B_i\delta\theta \qquad (2\text{-}34)$$

To find the maximum likelihood estimate we now want to find the $\delta\hat{W}$ and $\delta\theta$ that minimize $\tilde{\Phi}_i$ while at the same time satisfies the structural model constraints $\tilde{G}_i = 0$. To do this we introduce the vector of Lagrange multipliers λ and look for the stationary point of the Lagrangian:

$$\Lambda_i(\delta\hat{W},\ \delta\theta,\ \lambda_i) \equiv \tilde{\Phi}_i(\delta\hat{W}) + \lambda_i^T\tilde{G}_i \qquad (2\text{-}35)$$

Forming the normal equations of the Lagrangian and using equations (2-33) and (2-34) we have:

$$\partial\Lambda_i(\delta\hat{W},\ \delta\theta,\ \lambda_i)/\partial(\delta\hat{W}) = q_i + H_i\delta\hat{W} + A_i^T\lambda_i = 0 \qquad (2\text{-}36)$$

$$\partial\Lambda_i(\delta\hat{W},\ \delta\theta,\ \lambda_i)/\partial\lambda_i = G_i + A_i\delta\hat{W} + B_i\delta\theta = 0 \qquad (2\text{-}37)$$

$$\partial\Lambda_i(\delta\hat{W},\ \delta\theta,\ \lambda_i)/\partial(\delta\theta) = B_i^T\lambda_i = 0 \qquad (2\text{-}38)$$

From equation (2-36):

$$\delta\hat{W} = -H_i^{-1}\cdot(q_i + A_i^T\lambda_i) \qquad (2\text{-}39)$$

Substituting equation (2-39) into equation (2-37) we have:

$$G_i - A_iH_i^{-1}q_i - A_iH_i^{-1}A_i^T\lambda_i + B_i\,\delta\theta = 0 \qquad (2\text{-}40)$$

Solving for λ_i:

$$\lambda_i = (A_iH_i^{-1}A_i^T)^{-1}\cdot(B_i\,\delta\theta - A_iH_i^{-1}q_i + G_i) \qquad (2\text{-}41)$$

Now substituting equation (2-41) into equation (2-38):

$$B_i^T \cdot (A_i H_i^{-1} A_i)^{-1} \cdot (B_i \delta\theta - A_i H_i^{-1} q_i + G_i) = 0 \tag{2-42}$$

Solving for $\delta\theta$ we are left with:

$$\delta\theta = (B_i^T \cdot (A_i H_i^{-1} A_i^T)^{-1} \cdot B_i)^{-1} \cdot B_i^T \cdot (A_i H_i^{-1} A_i^T)^{-1} \cdot (A_i H_i^{-1} q_i - G_i) \tag{2-43}$$

We can define a new term C_i as:

$$C_i \equiv A_i H_i^{-1} A_i^T \tag{2-44}$$

and a term D_i as:

$$D_i \equiv B_i^T C_i^{-1} B_i \tag{2-45}$$

Substituting equations (2-44) and (2-45) into equation (2-43) gives:

$$\delta\theta = D_i^T B_i^T C_i^{-1} \cdot (A_i H_i^{-1} q_i - G_i) \tag{2-46}$$

Once we have calculated $\delta\theta$, we use equation (2-41) to calculate the Lagrangian multipliers which in turn are substituted into equation (2-39) to calculate the updates to the true measurements. Equations (2-39) and (2-46) then allow us to iteratively calculate the estimated beacon location and the estimated true values of the measurements with the recursion relations:

$$\begin{aligned} \theta^*_{i+1} &= \theta^*_i + \delta\theta \\ \hat{W}^*_{i+1} &= \hat{W}^*_i + \delta\hat{W} \end{aligned} \tag{2-47}$$

All we need to start the calculations are initial values for θ^* and for $\hat{W}^*$. Since the measurement errors are presumably small, we can use the actual measured values **W** for the initial values of $\hat{W}^*$. However, since we have no idea of the location of the acoustic beacon, the choice of an initial value for θ^* is more difficult. In the absence of any other information, we can choose the center of the search area for the initial value of θ^*.

In developing the solution to the maximum likelihood estimate we made the assumption that the measurement errors are normally distributed and independent between experiments. To determine if this is valid we refer back to the source of the measurement errors. The measurement error $\tilde{t}_{s\mu}$ is not normally distributed. However, if we do not make the assumption of a normally distributed error, and stay with a uniform distribution of errors, we will be unable to use the maximum likelihood method. This is because likelihood methods use gradient techniques to maximize a likelihood function which is based upon the probability distribution function of the errors. If the underlying probability distribution function is not smooth and continuous, there will not be a unique maximum, and a useful estimate can not be found. This means if we stay with the uniform distribution we will be forced to use a more computationally expensive direct search of the parameter space, or minimax method to estimate the acoustic beacon's location.

As a result of the uniformly distributed errors in $t_{s\mu}$, the solution that we obtain from the maximum likelihood estimator is suboptimal. There is also a chance that the method used to find the estimate that maximizes the likeliness function will diverge.

This happens when the estimate of the true value of $t_{s\mu}$ is further away from the actual true value of $t_{s\mu}$ then the quantization of the time base allows. To prevent the solution from diverging, we can place additional constraints upon the minimization of the objective function that confine the solution to be within a feasible region. We define the feasible region to be the region bounded by the inequality constraints:

$$
\begin{aligned}
-Range_{max} &\le x_T \le Range_{max} \\
-Range_{max} &\le y_T \le Range_{max} \\
-Depth_{max} &\le z_T \le 0
\end{aligned}
\tag{2-48}
$$

Where $Range_{max}$ defines the size of the search area that we are willing to scan, and $Depth_{max}$ is the maximum depth in the area of the search. The beacon's depth is bounded above by the sea surface level. If the solution to the maximum likelihood estimator is within the feasible region, we accept the estimate. If, however, the solution to the maximum likelihood estimator is outside the feasible region we must adjust the estimator to bring the solution inside the feasible region, or in the worst case, disregard the estimate.

To maintain a feasible solution we break equation (2-47) into component parts:

$$
\begin{aligned}
\theta^*_{i+1} &= \theta^*_i + \delta\theta = \theta^*_i + \rho_\theta U_\theta \\
\hat{W}^*_{i+1} &= \hat{W}^*_i + \delta\hat{W} = \hat{W}^*_i + \rho_{\hat{W}} U_{\hat{W}}
\end{aligned}
\tag{2-49}
$$

where U_θ and $U_{\hat{W}}$ are unit magnitude vectors that describe the direction of $\delta\theta$ and $\delta\hat{W}$, and ρ_θ and $\rho_{\hat{W}}$ are the magnitudes of $\delta\theta$ and $\delta\hat{W}$. We assume that $\delta\theta$, and hence U_θ, points in the proper direction to the maximum likelihood estimator. If

$\theta_i^* + \delta\theta$ is outside the feasible region defined by equation (2-48), we find the maximum value of ρ which satisfies:

$$\begin{bmatrix} -Range_{max} \\ -Range_{max} \\ Depth_{max} \end{bmatrix} < \theta_i^* + \rho_{max} U_\theta < \begin{bmatrix} Range_{max} \\ Range_{max} \\ 0 \end{bmatrix} \tag{2-50}$$

Once we have found ρ_{max} we replace ρ_θ and ρ_W in equation (2-49) which gives the new update equations:

$$\begin{aligned} \theta^*_{i+1} &= \theta^*_i + \frac{\rho_{max}}{\sqrt{2}} U_\theta \\ \hat{W}^*_{i+1} &= \hat{W}^*_i + \frac{\rho_{max}}{\sqrt{2}} U_{\hat{W}} \end{aligned} \tag{2-51}$$

This essentially shortens the step that we move in updating θ^* and $\mathbf{W}^*$ for the iteration. Notice that we do not simply replace ρ_θ and ρ_W with ρ_{max}. If we did this, θ^*_{i+1} would be on the edge of the feasible region. With θ^*_{i+1} on the edge of the feasible region, ρ_{max} will be zero, and θ^*_{i+1} will equal θ^*_i for all subsequent iterations. If this happens we will be unable to achieve any further updates of θ^*_{i+1} and $\mathbf{W}^*_i$. It is still possible that, after a certain number of iterations, θ^*_{i+1} lies on the edge of the feasible region. If this happens we must reject the estimate and try again using a different initial value for θ^* or more experimental data points.

2.6 Algorithm

Figure 2-4 at the end of this section describes the algorithm used to estimate the location of the acoustic beacon based on Deming's method as described by Bard.

There is no natural way of telling whether or not progress towards the solution has occurred in any given iteration when we use this method [Bard 1974]. Bard suggests calculating the quantities:

$$Z_o = \sum_{\mu=1}^{n} g_\mu(\hat{w}_{\mu_i}^*, \theta_i^*)^T c_{\mu_i}^{-1} g_\mu(\hat{w}_{\mu_i}^*, \theta_i^*) \qquad \text{(2-52)}$$

and:

$$Z_1 = \sum_{\mu=1}^{n} g_\mu(\hat{w}_{\mu_i}^*+\delta\hat{w}_\mu, \theta_i^*+\delta\theta)^T c_{\mu_i}^{-1} g_\mu(\hat{w}_{\mu_i}^*+\delta\hat{w}_\mu, \theta_i^*+\delta\theta) \qquad \text{(2-53)}$$

The rátional behind these quantities follows that at the true values of **W** and θ, G(**W**, θ) must equal zero. If Q_i is a positive definite matrix, we require that:

$$G(\hat{W}_{i+1}^*, \theta_{i+1}^*)^T Q_i\, G(\hat{W}_{i+1}^*, \theta_{i+1}^*) \le G(\hat{W}_i^*, \theta_i^*)^T Q_i\, G(\hat{W}_i^*, \theta_i^*) \qquad \text{(2-54)}$$

This means that as $\hat{W}_i^*$ and θ_i^* approach the true values of **W** and θ, the quantity $G(\hat{W}_i^*, \theta_i^*)^T Q_i\, G(\hat{W}_i^*, \theta_i^*)$ must decrease. The natural choice for Q_i is the inverse of the covariance matrix of $G(\hat{W}_i^*, \theta_i^*)$ [Bard 1974]. With V_μ as the covariance matrix of the measurements, the covariance matrix of $G(W_i^*, \theta_i^*)$ is approximately:

$$COV(G(\hat{W}_i^*, \theta_i^*)) \approx \sum_{\mu=1}^{n} a_{\mu_i} V_\mu a_{\mu_i}^T \qquad \text{(2-55)}$$

Now from equation (2-30) we have:

$$H_i = \partial^2 \Phi(\hat{W}^*) / \partial\hat{W} \partial\hat{W} = \begin{bmatrix} V_1^{-1} & 0 & \dots & 0 \\ 0 & V_2^{-1} & & \\ \vdots & & & \\ 0 & & \dots & V_n^{-1} \end{bmatrix} \qquad \text{(2-56)}$$

So we choose:

$$Q_i = COV(G(\hat{W}_i^*, \theta_i^*))^{-1} = \left(\sum_{\mu=1}^{n} a_{\mu_i} V_\mu a_{\mu_i} \right)^{-1} = \left(A_i H_i^{-1} A_i^T \right)^{-1} = C_i^{-1} \tag{2-57}$$

It then follows that: [Bard 1974]

$$G(\hat{W}_i^*, \theta_i^*)^T Q_i G(\hat{W}_i^*, \theta_i^*) = \sum_{\mu=1}^{n} g_\mu(\hat{w}_{\mu_i}^*, \theta_i^*)^T c_{\mu_i}^{-1} g_\mu(\hat{w}_{\mu_i}^*, \theta_i^*) \tag{2-58}$$

If Z_1 is less then Z_0, we accept $\delta\theta$ and $\delta\hat{W}$ and continue to the next iteration. If Z_1 is greater then Z_0, we shorten $\delta\theta$ and $\delta\hat{W}$ and then recalculate Z_1.

To determine when to stop the algorithm we follow the termination criterion of Marquardt [Bard 1974]. We define a tolerance:

$$\epsilon_i = 10^{-4} \cdot \left(\left| \theta_i^* \right|_{min} + 10^{-3} \right) \tag{2-59}$$

where $| \theta_i^* |_{min}$ is the smallest component of θ_i^*. The term 10^{-3} is added in case $| \theta_i^* |_{min}$ is close to zero. For each iteration we check to see if $| \delta\theta |_{min}$ is less then ε_i. If $| \delta\theta |_{min}$ is less then ε_i, we terminate the algorithm. If $| \delta\theta |_{min}$ is not less then ε_i, we continue. For each iteration we calculate a new tolerance value ε_i. This leads to a conservative termination criterion that will stop the algorithm if the parameter values cease changing.

To start the algorithm we choose an initial guess of the location of the

acoustic beacon, θ_o^*. We then use the measured values of **W** for the initial values of $\hat{\mathbf{W}}_o^*$. For each experiment μ, we calculate:

$$e_\mu = \hat{w}_\mu^* - w_\mu \tag{2-60}$$

$$a_\mu = \partial g_\mu / \partial \hat{w} \tag{2-61}$$

$$b_\mu = \partial g_\mu / \partial \theta \tag{2-62}$$

For the case of the stationary beacon a_μ is given by:

$$a_\mu = \begin{bmatrix} \dfrac{(\hat{x}_1^* - x_T^*)}{\sqrt{(\hat{x}_1^* - x_T^*)^2 + (\hat{y}_1^* - y_T^*)^2 + (z_T^*)^2}} \\ \dfrac{(\hat{y}_1^* - y_T^*)}{\sqrt{(\hat{x}_1^* - x_T^*)^2 + (\hat{y}_1^* - y_T^*)^2 + (z_T^*)^2}} \\ -C \\ \dfrac{(\hat{x}_2^* - x_T^*)}{\sqrt{(\hat{x}_2^* - x_T^*)^2 + (\hat{y}_2^* - y_T^*)^2 + (z_T^*)^2}} \\ -\dfrac{(\hat{y}_2^* - y_T^*)}{\sqrt{(\hat{x}_2^* - x_T^*)^2 + (\hat{y}_2^* - y_T^*)^2 + (z_T^*)^2}} \\ +C \end{bmatrix} \tag{2-63}$$

and b_μ is given by:

$$b_\mu = \begin{bmatrix} \dfrac{(\hat{x}_2^* - x_T^*)}{\sqrt{(\hat{x}_2^* - x_T^*)^2 + (\hat{y}_2^* - y_T^*)^2 + (z_T^*)^2}} - \dfrac{(\hat{x}_1^* - x_T^*)}{\sqrt{(\hat{x}_1^* - x_T^*)^2 + (\hat{y}_1^* - y_T^*)^2 + (z_T^*)^2}} \\ \dfrac{(\hat{y}_2^* - y_T^*)}{\sqrt{(\hat{x}_2^* - x_T^*)^2 + (\hat{y}_2^* - y_T^*)^2 + (z_T^*)^2}} - \dfrac{(\hat{y}_1^* - y_T^*)}{\sqrt{(\hat{x}_1^* - x_T^*)^2 + (\hat{y}_1^* - y_T^*)^2 + (z_T^*)^2}} \\ \dfrac{z_T^*}{\sqrt{(\hat{x}_1^* - x_T^*)^2 + (\hat{y}_1^* - y_T^*)^2 + (z_T^*)^2}} - \dfrac{z_T^*}{\sqrt{(\hat{x}_2^* - x_T^*)^2 + (\hat{y}_2^* - y_T^*)^2 + (z_T^*)^2}} \end{bmatrix} \tag{2-64}$$

Once we have a_μ and b_μ, we calculate:

$$c_\mu = a_\mu^T V_\mu a_\mu \tag{2-65}$$

$$Z_0 = \sum_{\mu=1}^{n} \frac{g(\hat{w}_\mu^*, \theta^*)^2}{c_\mu} \tag{2-66}$$

and:

$$D = \sum_{\mu=1}^{n} \left(\frac{1}{c_\mu}\right) \cdot b_\mu b_\mu^T \tag{2-67}$$

Using equation (2-46) we can solve for $\delta\theta$:

$$\delta\theta = D^{-1} \cdot \sum_{\mu=1}^{n} \left(\frac{1}{c_\mu}\right) \cdot \left(a_\mu^T e_\mu - g(\hat{w}_\mu^*, \theta^*)\right) \cdot b_\mu \tag{2-68}$$

Now using equation (2-41) we calculate the Lagrangian multipliers:

$$\lambda_\mu = \left(\frac{1}{c_\mu}\right) \cdot \left(b_\mu^T \delta\theta - a_\mu^T e_\mu + g(\hat{w}_\mu^*, \theta^*)\right) \tag{2-69}$$

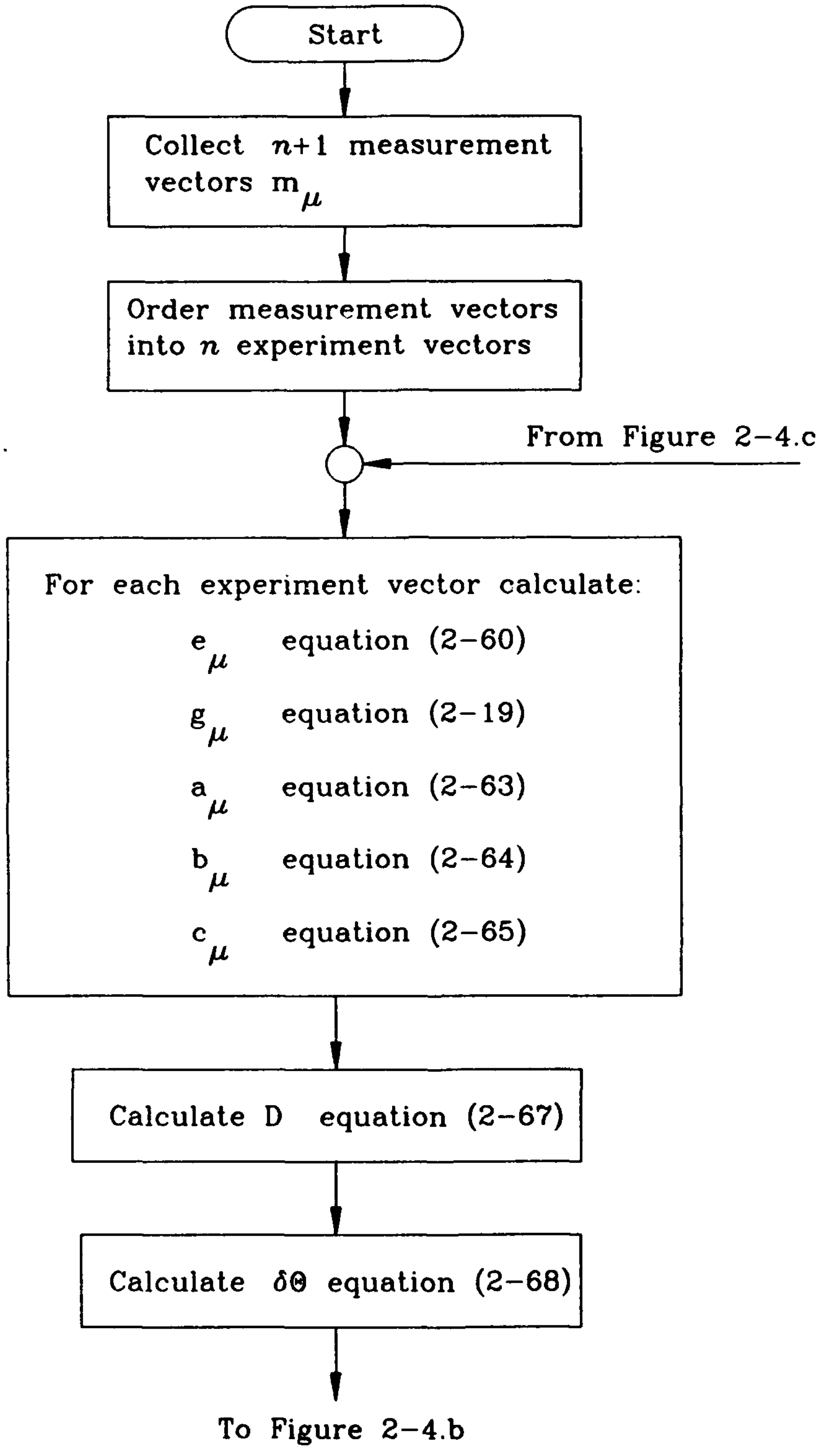
and finally we calculate:

$$\delta \hat{w}_{\mu} = -e_{\mu} - \lambda_{\mu} V_{\mu} a_{\mu} \tag{2-70}$$

Once we have $\delta\theta$ and $\delta\hat{\mathbf{W}}$ we check to see if $\theta_i^* + \delta\theta$ is in the feasible region. If $\theta_i^* + \delta\theta$ is not in the feasible region, we adjust the step size of $\delta\theta$ as describe earlier. Once we have a feasible step for $\delta\theta$, we calculate:

$$Z_1 = \sum_{\mu=1}^{n} \frac{g(\hat{w}_{\mu}^* + \delta\hat{w}_{\mu}, \theta^* + \delta\theta)^2}{c_{\mu}} \tag{2-71}$$

If Z_1 is less then Z_0, we accept the values of $\delta\theta$ and $\delta\hat{\mathbf{W}}$ and update θ^* and $\mathbf{W}^*$ using equation (2-47). We then move to the next iteration. If Z_1 is greater then Z_0, we assume that we have overshot the maximum likelihood estimate, shorten $\delta\theta$ and $\delta\hat{\mathbf{W}}$ by a factor of two, then recalculate Z_1. We continue this process until Z_1 is less then Z_0, or until $\delta\theta$ is negligible, at which point we have to stop the algorithm and begin again with a new initial value for θ^*, or more experimental data points.



Start
Collect $n+1$ measurement vectors m_μ
Order measurement vectors into n experiment vectors
From Figure 2-4.c
For each experiment vector calculate:
e_μ equation (2-60)
g_μ equation (2-19)
a_μ equation (2-63)
b_μ equation (2-64)
c_μ equation (2-65)
Calculate D equation (2-67)
Calculate $\delta\Theta$ equation (2-68)
To Figure 2-4.b

Figure 2-4.a

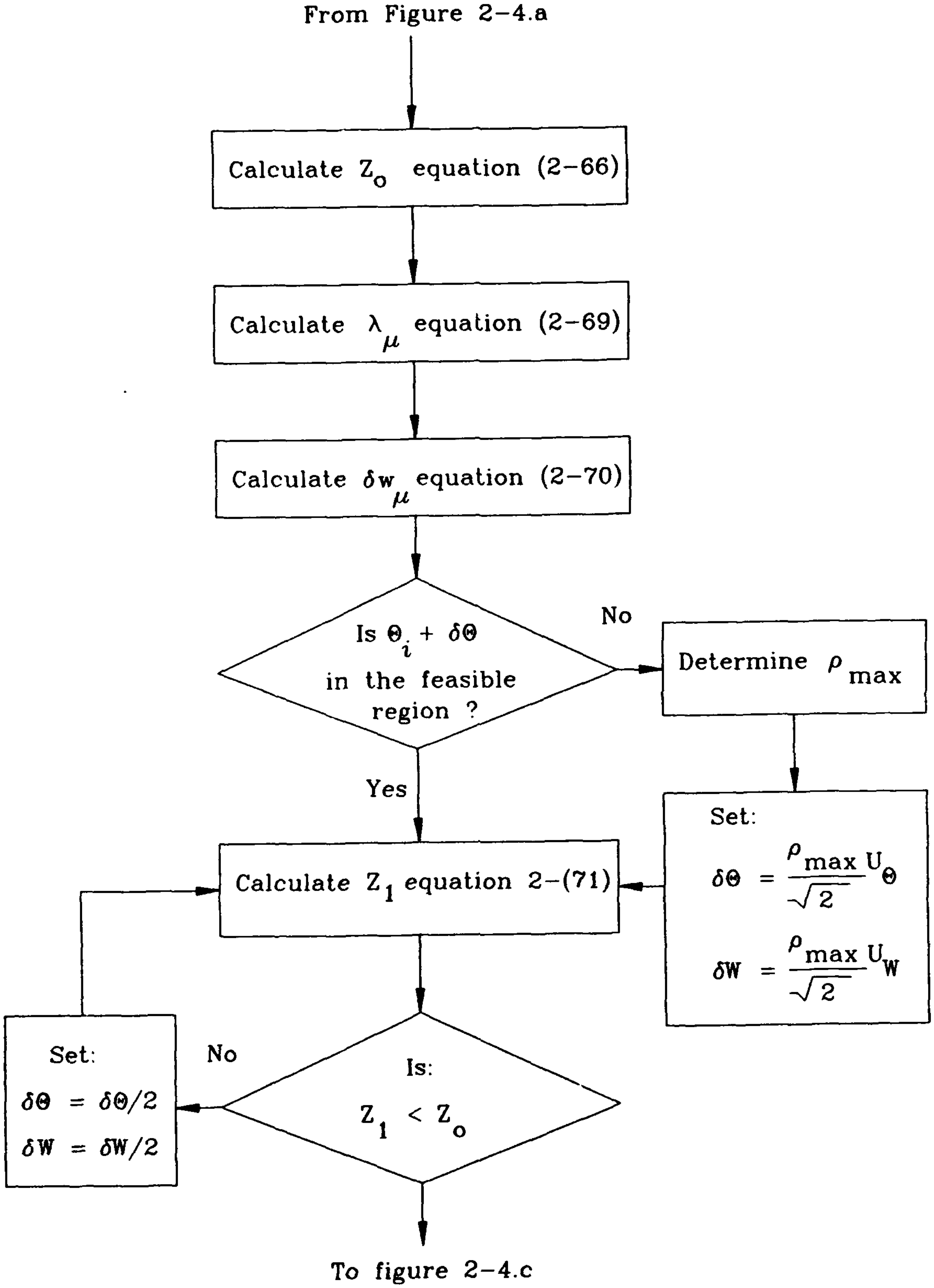
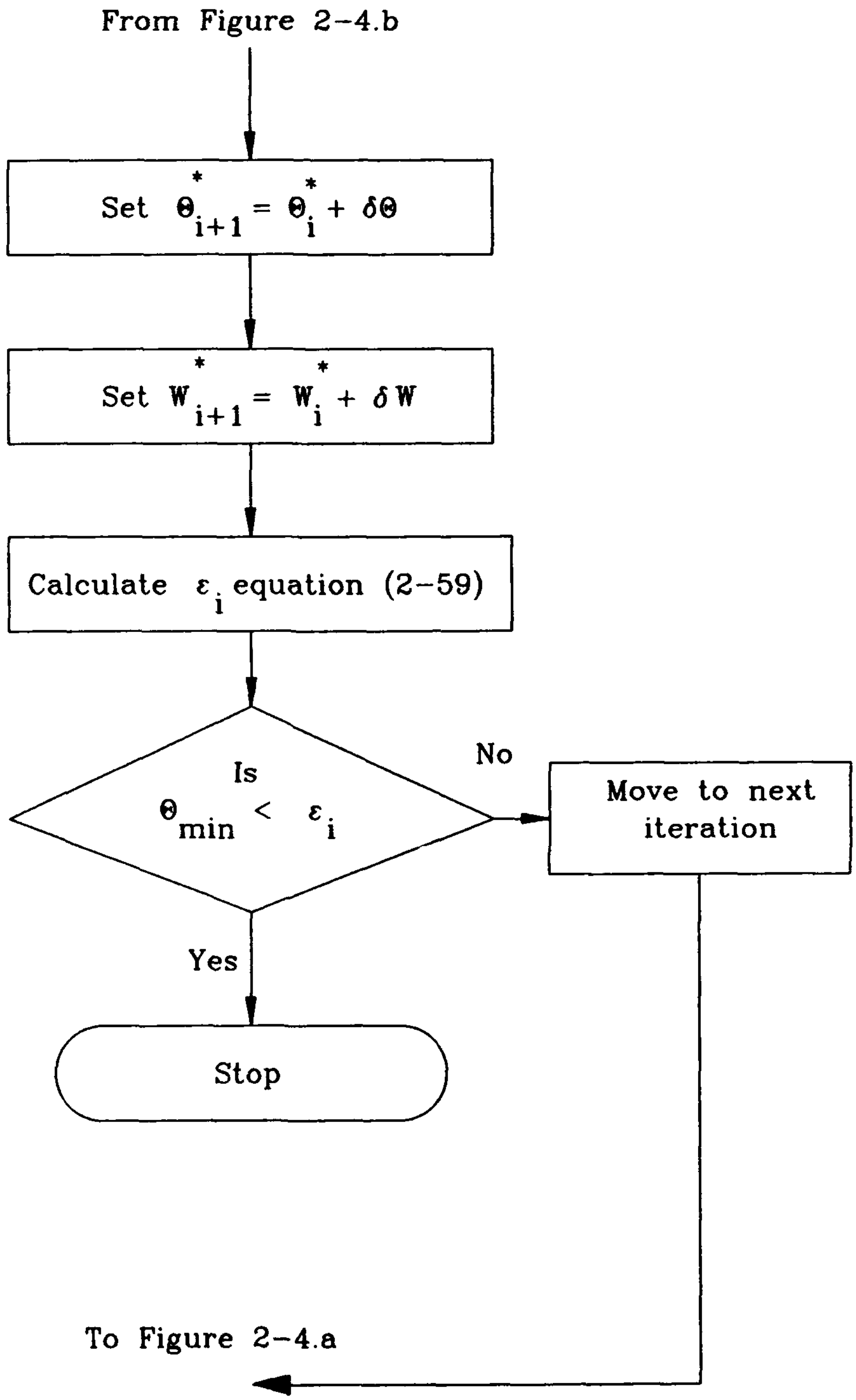

From Figure 2-4.a
Calculate Z_o equation (2-66)
Calculate λ_μ equation (2-69)
Calculate δw_μ equation (2-70)
Is $\Theta_i + \delta\Theta$ in the feasible region ?
No
Determine ρ_{max}
Yes
Set:
$\delta\Theta = \frac{\rho_{max} U_\Theta}{\sqrt{2}}$
$\delta W = \frac{\rho_{max} U_W}{\sqrt{2}}$
Calculate Z_1 equation 2-(71)
Set:
$\delta\Theta = \delta\Theta/2$
$\delta W = \delta W/2$
No
Is:
$Z_1 < Z_o$
To figure 2-4.c

Figure 2-4.b



From Figure 2-4.b
Set $\Theta^*_{i+1} = \Theta^*_i + \delta\Theta$
Set $W^*_{i+1} = W^*_i + \delta W$
Calculate ε_i equation (2-59)
Is $\Theta_{min} < \varepsilon_i$
No
Move to next iteration
Yes
Stop
To Figure 2-4.a

Figure 2-4.c

2.7 Accuracy of the Estimation

One of the benefits of using Deming's method for the maximum likelihood estimate is that the covariance of the estimated parameters is approximated by the inverse of the matrix D defined by equation (2-45) [Bard 1974]. This lets us quickly determine if the estimation of the acoustic beacon's position is acceptable, or if more experimental data needs to be collected to improve the estimation. We are also able to test if the assumptions made regarding the covariance matrix of the measurement errors are valid by examining the final residual vectors. We define a moment matrix M as:

$$M = \sum_{\mu=1}^{n} e_{\mu}^{*} e_{\mu}^{*T} \tag{2-72}$$

where the e_{μ}^{*} 's are the final residual vectors. To estimate the covariance of the residuals we take:

$$\tilde{V} = \frac{6}{n-3} \cdot M \tag{2-73}$$

where n is the number of experiment vectors used in the maximum likelihood estimation. Because of the nature of the constrained minimization problem, the first term in equation (2-73) is needed to correct for bias [Bard 1974]. We can compare the outcome of equation (2-73) to the assumed covariance matrix V_{μ} defined by equation (2-17) to see if the estimated covariance matrix V_{μ} is valid.

2.8 Summary

In this chapter we have developed the maximum likelihood estimate of the acoustic beacon's position, and a means in which to gauge the accuracy of the estimation. To estimate the beacon's position we defined a structural model that requires the observer to measure the time of receipt and location of receipt of two different signal from the acoustic beacon. The choice of how far apart the two signals that define an experiment vector (w_μ) must be, and the number of experiment vectors to use in the parameter estimation problem (n) remain as free parameters. In chapter IV we examine the performance of the algorithm of this chapter for different choices of these two free parameters.

Chapter III

Moving Acoustic Beacon

3.1 Introduction

In chapter II we developed the maximum likelihood estimator for a stationary acoustic beacon. In this chapter we extend the solution to include a moving acoustic beacon. The task of localizing an arbitrarily moving acoustic beacon is quite difficult. Abrupt changes in the acoustic beacon's velocity are difficult to parameterize and model. Therefore, we concentrate on the case of an acoustic beacon which moves in a constant velocity, linear path. The assumption of constant velocity, linear motion is a reasonable assumption for acoustic beacons located on instruments ascending at terminal velocity from the bottom, or for acoustic beacons located on instruments moving with an ocean current that is steady in the area of interest. With the assumption of constant linear motion, we modify the structural model given by equation (2-19) to include parameters that reflect the acoustic beacon's velocity. Once the new structural model is defined, we show that the solution to the maximum likelihood estimate for the moving beacon follows directly from the solution to the maximum likelihood estimate for the stationary acoustic beacon. Next, the algorithm developed to solve the stationary acoustic beacon localization problem is extended using the new structural model so that the position and velocity parameters of the moving acoustic beacon can be determined. Finally, the accuracy of the estimator is examined.

3.2 Problem Geometry

Consider an acoustic beacon having constant velocity components given by v_x, v_y, and v_z. We define a new parameter vector:

$$\theta = [\, x_o \quad y_o \quad z_o \quad v_x \quad v_y \quad v_z \,]^T \tag{3-1}$$

where the coordinates x_o, y_o, and z_o define the acoustic beacon's initial position on the same Cartesian coordinate system that was described in chapter II. For convenience, we assume the velocity components v_x, v_y, and v_z have the units of meters per second. The position of the acoustic beacon at an arbitrary time t_β can by found using:

$$\begin{bmatrix} x_T \\ y_T \\ z_T \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} + (t_\beta - T_o) \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{3-2}$$

where T_o is the reference time that corresponds to the acoustic beacon's initial position x_o, y_o, z_o.

As in chapter II, we define a measurement vector m_μ as:

$$m_\mu = [\, x_\mu \quad y_\mu \quad t_{s\mu} \,]^T \tag{3-3}$$

Where x_μ and y_μ are the ship's coordinates, in meters, on the same local Cartesian coordinate system defined for the acoustic beacon, and $t_{s\mu}$ is the synchronized time of receipt that corresponds to the observer's position x_μ, y_μ.

3.3 Modified Structural Model

To account for the linearly moving acoustic beacon, we redefine the function $\mathbf{f}(\mathbf{m}_\mu, \theta)$ from chapter II as:

$$f(m_\mu, \theta) \equiv \sqrt{[x_\mu - x_o - (t_\beta - T_o)v_x]^2 + [y_\mu - y_o - (t_\beta - T_o)v_y]^2 + [z_o + (t_\beta - T_o)v_z]^2} \\ - \tau(C, t_{s\mu}) \tag{3-4}$$

where t_β represents the time that the acoustic beacon transmitted the signal that the observer received at the synchronized time of receipt $t_{s\mu}$. Using the same homogeneous ocean model, and direct path propagation from chapter II, we simplify equation (3-4) to:

$$f(m_\mu, \theta) \equiv \sqrt{[x_\mu - x_o - (t_\beta - T_o)v_x]^2 + [y_\mu - y_o - (t_\beta - T_o)v_y]^2 + [z_o + (t_\beta - T_o)v_z]^2} \\ - C(t_{s\mu} - T_o) \tag{3-5}$$

Now assume that at time T_o the acoustic beacon transmitted the first signal that the observer receives. To find when the current signal received at the synchronized time of receipt $t_{s\mu}$ was transmitted, we note that the acoustic beacon transmits a signal for every pulse repetition interval. With T_o as a reference, t_β can be found by:

$$t_\beta = T_o + i \cdot T_p \tag{3-6}$$

where T_p is the pulse repetition interval of the acoustic beacon and i is number of pulse repetition intervals that separate the signal transmitted at time t_β from the signal transmitted at reference time T_o. To determine i, the observer counts how many signals are received between the first received signal and the signal received at the

$$g(\hat{w}_\mu, \theta) = \sqrt{(\hat{x}_1 - x_o - i \cdot T_p \cdot v_x)^2 + (\hat{y}_1 - y_o - i \cdot T_p \cdot v_y)^2 + (z_o + i \cdot T_p \cdot v_z)^2}$$
$$- \sqrt{(\hat{x}_2 - x_o - j \cdot T_p \cdot v_x)^2 + (\hat{y}_2 - y_o - j \cdot T_p \cdot v_y)^2 + (z_o + j \cdot T_p \cdot v_z)^2} \qquad \text{(3-10)}$$
$$- C \cdot (\hat{t}_{s1} - \hat{t}_{s2})$$

This is analogous to equation (2-19) with additional terms included to correct for the beacon's motion.

3.4 Maximum Likelihood Estimator Solution

The solution to the maximum likelihood estimation problem for the case of the moving acoustic beacon follows directly from the solution to the maximum likelihood estimation problem for the case of the stationary acoustic beacon. Since only the structural model has changed, the likelihood function log (L ($\hat{W}$, V)), the Lagrangian function $\Lambda(\hat{W}, \theta, V, \lambda_1, \ldots, \lambda_n)$ and the objective function $\Phi(W_i^*)$ all have the same form as the corresponding functions defined for the stationary beacon. To solve for the maximum likelihood estimator of the parameter vector θ, we simply substitute the modified structural model given by equation (3-10) for the stationary structural mode given by equation (2-19), and follow the method of Deming described in chapter II. There are, however, a few differences that must be accounted for.

If the measurement errors are the same for both cases, we still expect the solution to the maximum likelihood estimator to be suboptimal. Because of the uniform distribution of the measurement errors $\tilde{t}_{s\mu}$, we must continue to confine the solution of the maximum likelihood estimation problem to a feasible region. With the addition of the parameters v_x, v_y, and v_z to θ, we expand the feasible region defined by

equation (2-48) to:

$$
\begin{aligned}
-Range_{max} &\le x_o \le Range_{max} \\
-Range_{max} &\le y_o \le Range_{max} \\
-Depth_{max} &\le z_o \le 0 \\
-v_{x_{max}} &\le v_x \le v_{x_{max}} \\
-v_{y_{max}} &\le v_y \le v_{y_{max}} \\
-v_{z_{max}} &\le v_z \le v_{z_{max}}
\end{aligned}
\tag{3-11}
$$

As in chapter II, to maintain a feasible solution, we break $\delta\theta$ into component parts:

$$\theta^*_{i+1} = \theta^*_i + \delta\theta = \theta^*_i + \rho_\theta U_\theta \tag{3-12}$$

where U_θ is unit magnitude vector that describes the direction of $\delta\theta$, and ρ_θ is the magnitude of $\delta\theta$. We assume that $\delta\theta$, and therefore U_θ , points in the proper direction to the maximum likelihood estimator. If $\theta^*_i + \delta\theta$ is outside the feasible region defined by equation (3-11), we find the maximum value of ρ that satisfies:

$$
\begin{bmatrix} -Range_{max} \\ -Range_{max} \\ -Depth_{max} \\ -v_{x_{max}} \\ -v_{y_{max}} \\ -v_{z_{max}} \end{bmatrix}
< \theta^*_i + \rho_{max} U_\theta <
\begin{bmatrix} Range_{max} \\ Range_{max} \\ 0 \\ v_{x_{max}} \\ v_{y_{max}} \\ v_{z_{max}} \end{bmatrix}
\tag{3-13}
$$

Once we have found ρ_{max}, we define the new recursion relations:

$$\theta^*_{i+1} = \theta^*_i + \frac{\rho_{max}}{\sqrt{2}} U_\theta$$

$$\hat{W}^*_{i+1} = \hat{W}^*_i + \frac{\rho_{max}}{\sqrt{2}} U_{\hat{W}} \tag{3-14}$$

As with the case of the stationary acoustic beacon, if, after a few iterations, θ^*_i lies on the edge of the feasible region, ρ_{max} will be zero and θ^*_{i+1} will equal θ^*_i for all subsequent iterations. If this happens we reject the estimate, and try again using a different initial value for θ^*, or more experimental data points.

3.5 Algorithm

Figure 3-1 at the end of this section describes the algorithm used to solve for the acoustic beacon's position and velocity. The algorithm is essentially the same algorithm developed in chapter II with changes made to account for the modified structural model of equation (3-10). Because of the modified structural model, we must reevaluate the quantities a_μ and b_μ. From equation (3-10), we define the quantities:

$$d_1 = \sqrt{(\hat{x}_1^* - x_o^* - i \cdot T_p v_x^*)^2 + (\hat{y}_1^* - y_o^* - i \cdot T_p v_y^*)^2 + (z_o^* + i \cdot T_p v_z^*)^2} \tag{3-15}$$

$$d_2 = \sqrt{(\hat{x}_2^* - x_o^* - j \cdot T_p \cdot v_x^*)^2 + (\hat{y}_2^* - y_o^* - j \cdot T_p \cdot v_y^*)^2 + (z_o^* + j \cdot T_p \cdot v_z^*)^2} \tag{3-16}$$

a_μ and b_μ are then given by:

$$a_\mu = \partial g(\hat{w}_\mu^*, \theta^*)/\partial \hat{w}_\mu = \begin{bmatrix} \dfrac{(\hat{x}_1^* - x_o^* - i \cdot T_p \cdot v_x^*)}{d_1} \\ \dfrac{(\hat{y}_1^* - y_o^* - i \cdot T_p \cdot v_y^*)}{d_1} \\ -C \\ \dfrac{(\hat{x}_2^* - x_o^* - j \cdot T_p \cdot v_x^*)}{d_2} \\ \dfrac{(\hat{y}_2^* - y_o^* - j \cdot T_p \cdot v_y^*)}{d_2} \\ C \end{bmatrix} \tag{3-17}$$

$$b_\mu = \partial g(\hat{w}_\mu^*, \theta^*)/\partial \theta$$

$$= \begin{bmatrix} \left(\dfrac{(\hat{x}_2^* - x_o^* - j \cdot T_p \cdot v_x^*)}{d_2} - \dfrac{(\hat{x}_1^* - x_o^* - i \cdot T_p \cdot v_x^*)}{d_1} \right) \\ \left(\dfrac{(\hat{y}_2^* - y_o^* - j \cdot T_p \cdot v_y^*)}{d_2} - \dfrac{(\hat{y}_1^* - y_o - i \cdot T_p \cdot v_y^*)}{d_1} \right) \\ \left(\dfrac{(z_o^* + i \cdot T_p \cdot v_z^*)}{d_1} - \dfrac{(z_o^* + j \cdot T_p v_x^*)}{d_2} \right) \\ \left(\dfrac{j \cdot T_p(\hat{x}_2^* - x_o^* - j \cdot T_p \cdot v_x^*)}{d_2} - \dfrac{i \cdot T_p(\hat{x}_1^* - x_o^* - i \cdot T_p \cdot v_x^*)}{d_1} \right) \\ \left(\dfrac{j \cdot T_p(\hat{y}_2^* - y_o^* - j \cdot T_p \cdot v_y^*)}{d_2} - \dfrac{i \cdot T_p(\hat{y}_1^* - y_o^* - i \cdot T_p \cdot v_y^*)}{d_1} \right) \\ \left(\dfrac{i \cdot T_p(z_o^* + i \cdot T_p \cdot v_z^*)}{d_1} - \dfrac{j \cdot T_p(z_o^* + j \cdot T_p \cdot v_x^*)}{d_2} \right) \end{bmatrix} \tag{3-18}$$

To evaluate the structural model, and the terms a_μ and b_μ ,we need to know the values of i and j that represent the time that the received signals were transmitted. This forces the observer to keep a count of the number of signals that are received, with the first received signal corresponding to $i = 0$.

Once we have found the maximum likelihood estimate of the parameter vector θ, we need to evaluate the estimated current position of the acoustic beacon. The position defined by x_o^*, y_o^*, z_o^* is the estimated initial position of the acoustic beacon at the time T_o. We assume that the time T_o corresponds to the case $i = 0$. To determine the current position of the acoustic beacon, we project the solution forward along the path described by the estimated velocity components v_x^*, v_y^*, v_z^*. The estimated current position of the acoustic beacon is then given by:

$$\begin{bmatrix} x_T^* \\ y_T^* \\ z_T^* \end{bmatrix} = \begin{bmatrix} x_o^* \\ y_o^* \\ z_o^* \end{bmatrix} + i \cdot T_p \begin{bmatrix} v_x^* \\ v_y^* \\ v_z^* \end{bmatrix} \tag{3-19}$$

where i is the total number of beacon signals received and T_p is the pulse repetition interval of the acoustic beacon. The quantity $(i \cdot T_p)$ represents the time it took to collect the measurements used to calculate the maximum likelihood estimator.

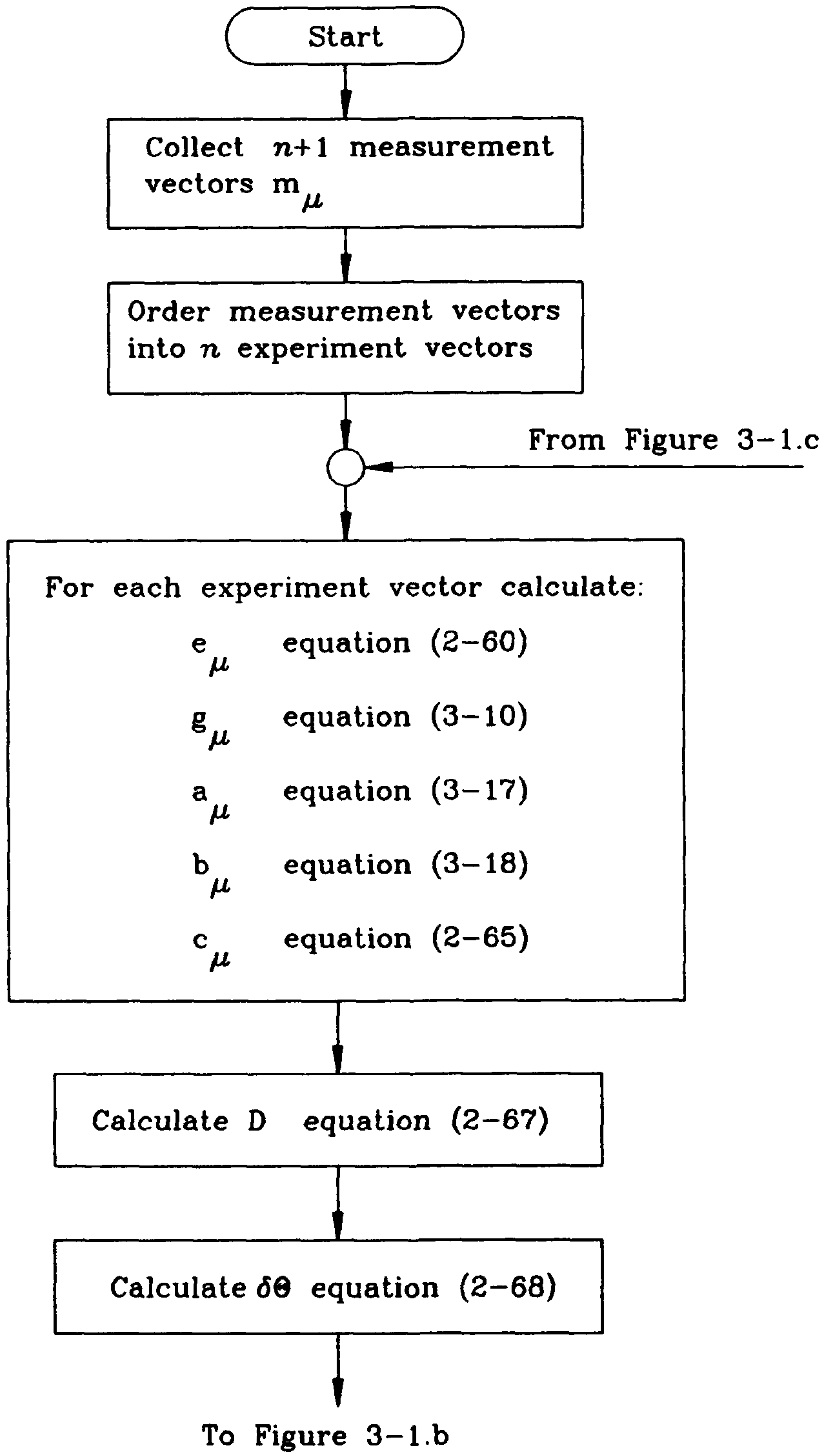

Start
Collect $n+1$ measurement vectors m_μ
Order measurement vectors into n experiment vectors
From Figure 3-1.c
For each experiment vector calculate:
e_μ equation (2-60)
g_μ equation (3-10)
a_μ equation (3-17)
b_μ equation (3-18)
c_μ equation (2-65)
Calculate D equation (2-67)
Calculate $\delta\Theta$ equation (2-68)
To Figure 3-1.b

Figure 3-1.a

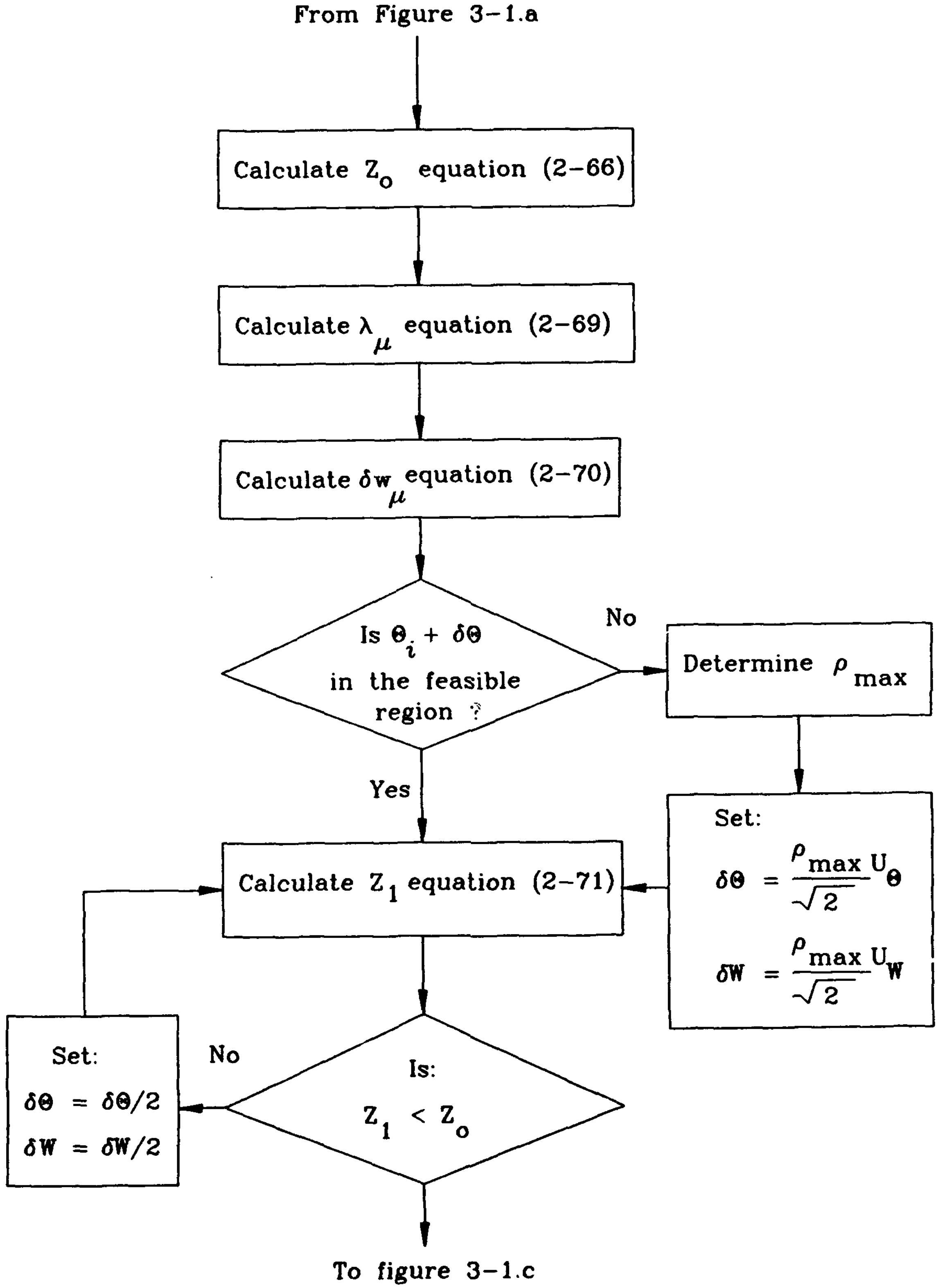

From Figure 3-1.a
Calculate Z_o equation (2-66)
Calculate λ_μ equation (2-69)
Calculate δw_μ equation (2-70)
Is $\Theta_i + \delta\Theta$ in the feasible region ?
No
Determine ρ_{max}
Yes
Set:
$\delta\Theta = \frac{\rho_{max} U_\Theta}{\sqrt{2}}$
$\delta W = \frac{\rho_{max} U_W}{\sqrt{2}}$
Calculate Z_1 equation (2-71)
Set:
$\delta\Theta = \delta\Theta/2$
$\delta W = \delta W/2$
No
Is:
$Z_1 < Z_o$
To figure 3-1.c

Figure 3-1.b

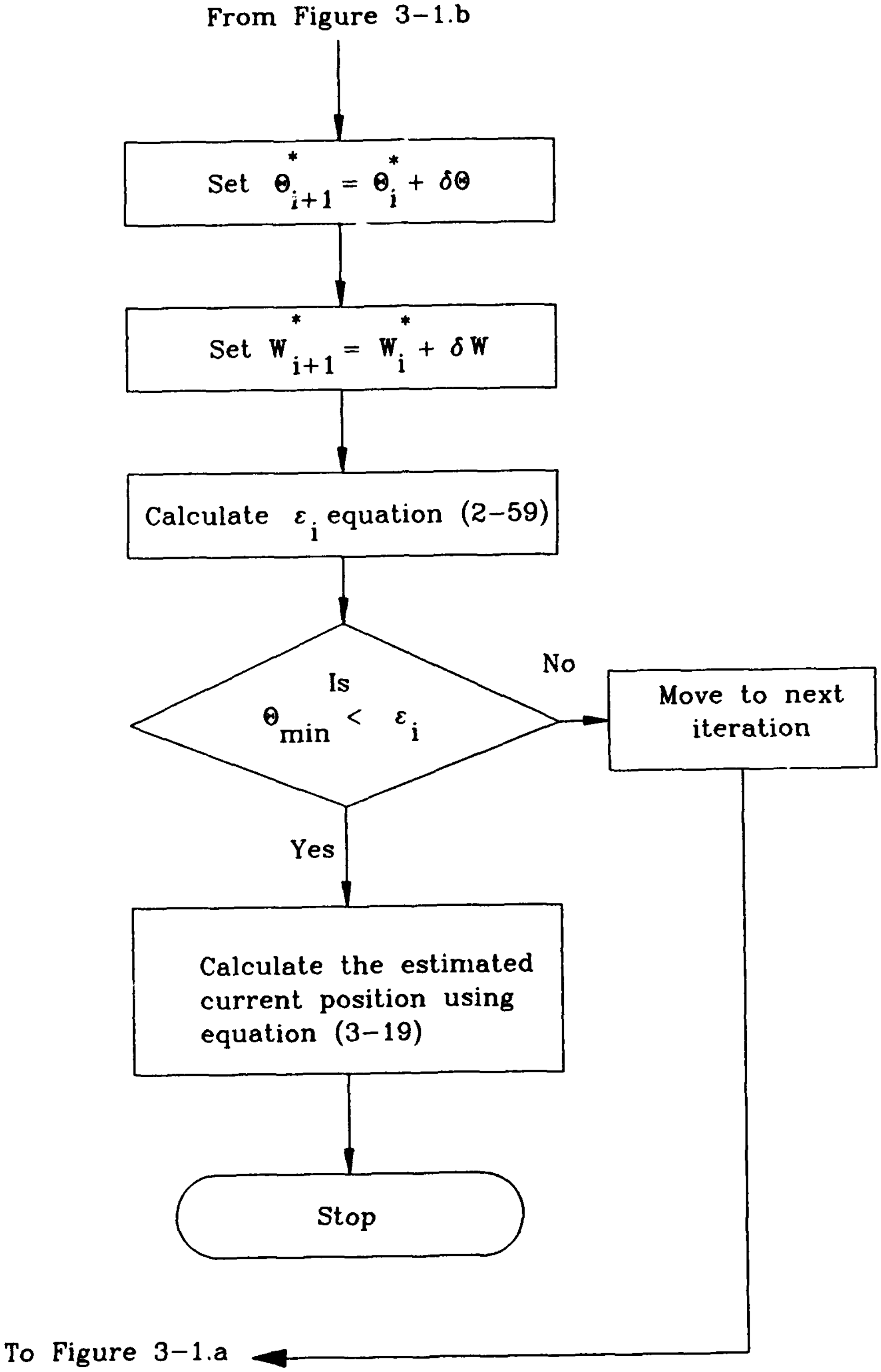

From Figure 3-1.b
Set $\Theta^*_{i+1} = \Theta^*_i + \delta\Theta$
Set $W^*_{i+1} = W^*_i + \delta W$
Calculate ε_i equation (2-59)
Is $\Theta_{min} < \varepsilon_i$
No
Move to next iteration
Yes
Calculate the estimated current position using equation (3-19)
Stop
To Figure 3-1.a

Figure 3-1.c

3.6 Accuracy of the Estimator

To estimate the accuracy of the solution to the maximum likelihood estimator, we examine the estimated covariance matrix of the parameter vector θ. As with the stationary beacon, the estimated covariance matrix of the parameter vector θ is given by the inverse of the matrix D or:

$$V_\theta^* = (\, B^T \, C^{-1} \, B \,)^{-1} = \left(\sum_{\mu=1}^{n} \left(\frac{1}{c_\mu} \right) \cdot b_\mu \; b_\mu \right)^{-1} \tag{3-20}$$

The variance of the velocity components v_x^*, v_y^*, and v_z^* are the last three diagonal elements of the covariance matrix V_θ^*.

To determine the variances of the current acoustic beacon position estimates we calculate:

$$\sigma_{x_T}^{\;2} = E\left[\left((x_o + i \cdot T_p \, v_x) - E\left[x_o + i \cdot T_p \, v_x \right] \right)^2 \right] \tag{3-21}$$

$$\sigma_{y_T}^{\;2} = E\left[\left((y_o + i \cdot T_p \, v_y) - E\left[y_o + i \cdot T_p \, v_y \right] \right)^2 \right] \tag{3-22}$$

$$\sigma_{z_T}^{\;2} = E\left[\left((z_o + i \cdot T_p \, v_z) - E\left[z_o + i \cdot T_p \, v_z \right] \right)^2 \right] \tag{3-23}$$

Simplifying equations (3-21) through (3-23) yields:

$$\begin{aligned} \sigma_{x_T}^{\;2} = {} & \sigma_{x_o}^{\;2} + \left(i \cdot T_p \right)^2 \cdot \sigma_{v_x}^{\;2} + 2 \cdot \left(i \cdot T_p \right) E\left[x_o v_x \right] \\ & - E\left[x_o \right]^2 - 2 \cdot \left(i \cdot T_p \right) E\left[v_x \right] E\left[x_o \right] - \left(i \cdot T_p \cdot E\left[v_x \right] \right)^2 \end{aligned} \tag{3-24}$$

$$\begin{aligned} \sigma_{y_T}^{\;2} = {} & \sigma_{y_o}^{\;2} + \left(i \cdot T_p \right)^2 \cdot \sigma_{v_y}^{\;2} + 2 \cdot \left(i \cdot T_p \right) E\left[y_o v_y \right] \\ & - E\left[y_o \right]^2 - 2 \cdot \left(i \cdot T_p \right) E\left[v_y \right] E\left[y_o \right] - \left(i \cdot T_p \cdot E\left[v_y \right] \right)^2 \end{aligned} \tag{3-25}$$

$$\sigma_{z_T}^{\;2} = \sigma_{z_o}^{\;2} + \left(i\cdot T_p\right)^2\cdot\sigma_{v_z}^{\;2} + 2\cdot\left(i\cdot T_p\right)E\left[z_o v_z\right]$$
$$- E\left[z_o\right]^2 - 2\cdot\left(i\cdot T_p\right)E\left[v_z\right]E\left[z_o\right] - \left(i\cdot T_p\cdot E\left[v_z\right]\right)^2 \qquad \textbf{(3-26)}$$

Now if we assume that the estimate of the parameter vector θ is unbiased, we can substitute the estimated parameter values for their means in equations (3-24) through (3-26). This gives us a method in which to calculate the estimated variance of the projected current position:

$$\sigma_{x_T}^{\;2} = \sigma_{x_o}^{\;2} + \left(i\cdot T_p\right)^2\cdot\sigma_{v_x}^{\;2} + 2\cdot\left(i\cdot T_p\right)E\left[x_o v_x\right]$$
$$- (x_o^*)^2 - 2\cdot\left(i\cdot T_p\right)v_x^* x_o^* - \left(i\cdot T_p\cdot v_x^*\right)^2 \qquad \textbf{(3-27)}$$

$$\sigma_{y_T}^{\;2} = \sigma_{y_o}^{\;2} + \left(i\cdot T_p\right)^2\cdot\sigma_{v_y}^{\;2} + 2\cdot\left(i\cdot T_p\right)E\left[y_o v_y\right]$$
$$- (y_o^*)^2 - 2\cdot\left(i\cdot T_p\right)v_y y_o^* - \left(i\cdot T_p\cdot v_y^*\right)^2 \qquad \textbf{(3-28)}$$

$$\sigma_{z_T}^{\;2} = \sigma_{z_o}^{\;2} + \left(i\cdot T_p\right)^2\cdot\sigma_{v_z}^{\;2} + 2\cdot\left(i\cdot T_p\right)E\left[z_o v_z\right]$$
$$- (z_o^*)^2 - 2\cdot\left(i\cdot T_p\right)v_z^* z_o^* - \left(i\cdot T_p\cdot v_z^*\right)^2 \qquad \textbf{(3-39)}$$

From equations (3-27) through (3-29) we see that the variance of the estimated current position has a complex relationship to the number of experiment vectors used and the effective sensor spacing. As these quantities increase, we expect that the variances of the estimated initial positions and velocities will decrease. However, as the number of experiment vectors used and the effective sensor spacing increases, the term *i* will increase and the variance of the estimate of the current position of the acoustic beacon can also increase. This is because if we project the initial position of the acoustic beacon forward based on the estimated velocity parameters, any errors in the acoustic

beacon's velocity causes us to move the estimate in the wrong direction. The choice of how many experiment points to collect, or how long we spend collecting data points, will be examined in chapter IV.

As with the stationary beacon, we can test the assumptions made regarding the covariance matrix of the measurement errors. We define the moment matrix as:

$$M = \sum_{\mu=1}^{n} e_{\mu}^{*} e_{\mu}^{*T} \tag{3-30}$$

where the e_{μ}^{*} 's are the final residual vectors defined as in chapter II. To estimate the covariance of the residuals we take:

$$\tilde{V} = \frac{6}{n-6} \cdot M \tag{3-31}$$

where n is the number of experiment vectors used in the maximum likelihood estimation. The first term in equation (3-31) corrects for the bias caused by the constrained minimization problem [Bard 1974]. We can compare the results of equation (3-31) with the assumed covariance matrix of the measurement errors to judge if the assumed covariance matrix V_{μ} is valid.

3.7 Summary

In this chapter we have extended the results of chapter II to include linear motion of the acoustic beacon. We found that the solution to the maximum likelihood estimator for the case of the moving beacon follows the same form as the solution for the case of the stationary beacon. The quantity $(j - i)$ is one of the free parameters of the moving acoustic beacon estimation problem. We define this quantity as the

effective sensor spacing. This quantity corresponds to the spacing in time between the two signals used to form the experiment vector w_{μ}. The other free parameter is the number of experiment vectors to use in the parameter estimation problem. In the next chapter we examine the performance of the algorithm described in this chapter for different choices of the two free parameters.

Chapter IV

Algorithm Simulations and Tests

4.1 Introduction

In chapters II and III we developed algorithms to solve for the maximum likelihood estimate of an acoustic beacon's position for both the stationary and the moving beacon. In both chapters we were left with choosing how many experiment vectors (w_μ) to use, and which measurement vectors (m_μ) to use to define the experiment vectors. We defined the effective element spacing to be the number of pulse repetition intervals (T_p) between the measurement vectors that form the experiment vector w_μ. In this chapter we examine the performance of the algorithms developed in chapters II and III for different numbers of experiment vectors, and for different effective element spacings. We show that the accuracy of the estimated beacon position (θ^*) improves, while the variance of the estimated position decreases as the number of experiment vectors used to find the maximum likelihood estimate increases. We also show that the accuracy of the estimated beacon position improves, and the variance of the estimated position decreases with increased effective sensor spacing. This result is similar to the methods mentioned in chapter I.

To test the algorithms of chapters II and III, Monte Carlo simulations were conducted for several different initial positions and velocities of the acoustic beacon. Results from six of the scenarios tested are presented in the following sections. We assume that the measurement errors are as defined in section 2.4. To see how the accuracy of the measured position of receipt (x_μ, y_μ) affects the maximum likelihood

estimate, we use GPS errors of 3 meters rms and 25 meters rms. These errors approximate the errors of a Differential/P code GPS receiver and a C/A code GPS receiver respectively. For measurement errors in the synchronized time of receipt $t_{s\mu}$, we assume the sampling rate of the receiver, T_s, is 1 millisecond and use equation (2-16) to define the variance.

4.2 Synthetic Measurement Vector Generation

To test the algorithms of chapters II and III, synthetic measurement vectors were generated for several different acoustic beacon positions and velocities. Table 4-1 lists the initial position and velocities for each of the scenarios presented. We assume that the receiving ship moves in a hexagonal search path with legs of 1000 meters. This is a reasonable search path in the general location of the acoustic beacon which is easy to model. The actual path of the receiving ship is not critical,

Scenario	x_B (m)	y_B (m)	z_B (m)	v_x (m/s)	v_y (m/s)	v_z (m/s)
1	0	0	-2000	0	0	0
2	0	866	-2000	0	0	0
3	-1000	1000	-2000	0	0	0
4	0	0	-2000	0.5	-0.5	0.25
5	0	866	-2000	0.5	-0.5	0.25
6	-1000	1000	-2000	0.5	-0.5	0.25

Table 4-1

as long as it is not linear. If the receiving ship's path is linear we will be unable to

accurately resolve the acoustic beacon's position in three dimensions. This is analogous to the inability of a conventional linear array to resolve the direction of arrival of a signal in two dimensions. We restrict the speed of the receiving ship to five meters per second so that flow noise does not become a problem.

Figure 4-1 shows the positions of the acoustic beacon for the scenarios defined in Table 4-1 compared to the path of the receiving ship.

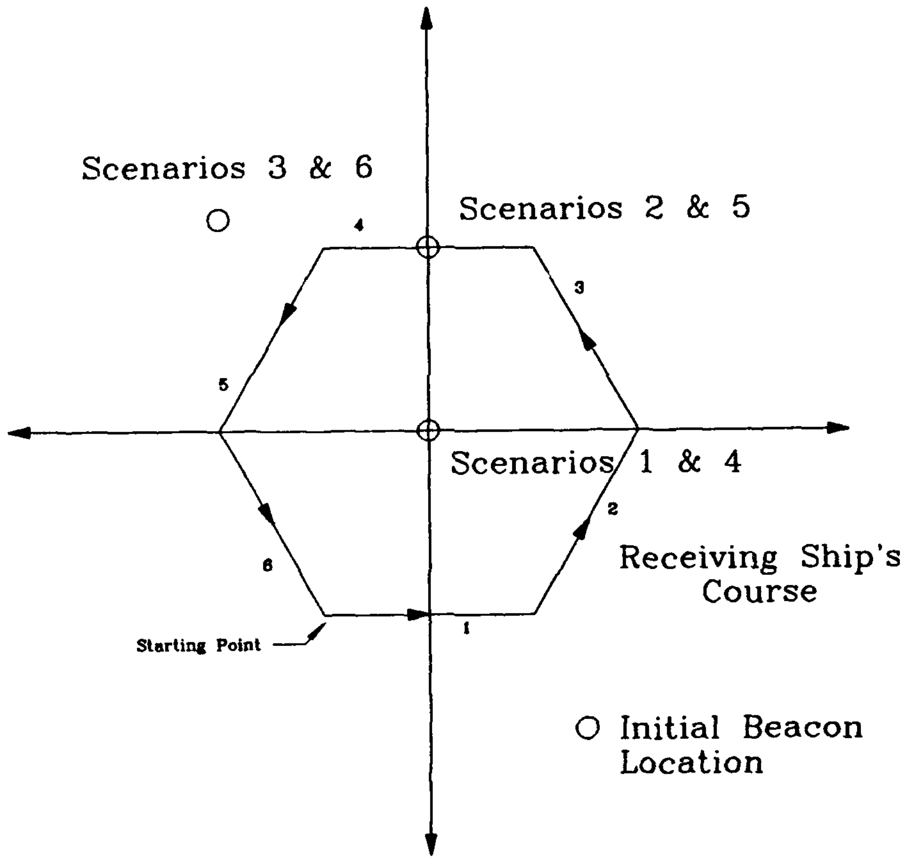


Figure 4-1

Scenarios 1 and 4 corresponds to an acoustic beacon located in the center of the search path, scenarios 2 and 5 correspond to an acoustic beacon located directly below the search path, and scenarios 3 and 6 correspond to an acoustic beacon located outside of

the search path.

To generate the synthetic measurement vectors m_μ, we solve for the intercept of the receiving ship and the beacon signal. Figure 4-2 shows the intercept geometry.

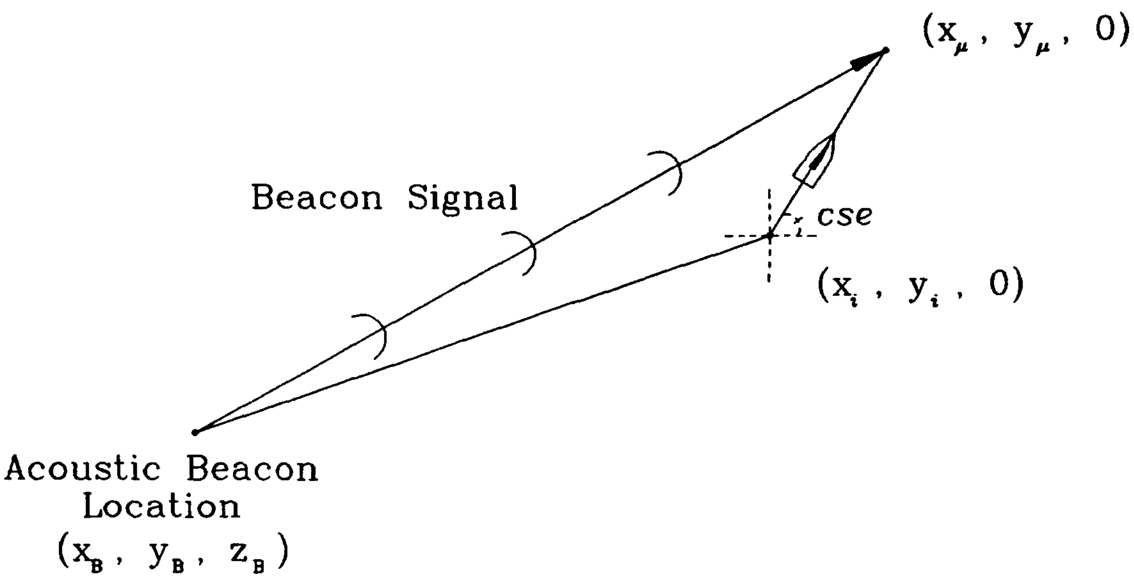


Figure 4-2

The position x_i, y_i is the location of the receiving ship when the beacon signal is transmitted, and the position x_μ, y_μ is the location of the ship when the beacon signal is received. Using a homogeneous ocean model and a direct path propagation mode, we calculate the synchronized time of receipt by finding the positive root of:

$$\alpha \cdot t^2 + \beta \cdot t + \gamma = 0 \tag{4-1}$$

where:

$$\begin{aligned} \alpha &= s^2 - C^2 \\ \beta &= 2 \cdot s \cdot [\cos(cse) \cdot (x_i - x_B) + \sin(cse) \cdot (y_i - y_B)] \\ \gamma &= (x_i - x_B)^2 + (y_i - y_B)^2 + z_B^2 \end{aligned} \tag{4-2}$$

Here s is the speed of the receiving ship in meters per second, C is the speed of sound in sea water, and cse is the course of the receiving ship in degrees. It is assumed that the receiving ship does not change its course for the duration of the calculation. Once we have the synchronized time of receipt, we calculate the position of receipt x_μ, y_μ using:

$$\begin{aligned} x_\mu &= x_l + t_{s\mu} \cdot s \cdot \cos(cse) \\ y_\mu &= y_l + t_{s\mu} \cdot s \cdot \sin(cse) \end{aligned} \tag{4-3}$$

To find the next synchronized time of receipt and position of receipt (x_μ, y_μ) we update the acoustic beacon's position based on its velocity:

$$\begin{aligned} x_B &= x_B + T_p \cdot v_x \\ y_B &= y_B + T_p \cdot v_y \\ z_B &= z_B + T_p \cdot v_z \end{aligned} \tag{4-4}$$

where T_p is the pulse repetition interval of the acoustic beacon. For the results presented in the following sections we assume the acoustic beacon has a pulse repetition interval of 2 seconds. Next we find the location of the receiving ship when the new beacon signal is transmitted using:

$$\begin{aligned} x_l &= x_l + T_p \cdot s \cdot \cos(cse) \\ y_l &= y_l + T_p \cdot s \cdot \sin(cse) \end{aligned} \tag{4-5}$$

We then solve for the intercept of the beacon signal and the receiving ship using equations (4-1) and (4-2). We continue in this manner until we have generated a sufficient number of measurement vectors m_μ. Once we have generated the

measurement vectors, we add normally distributed random error terms to each measurement based upon the errors discussed in section 2.4.

4.3 Stationary Acoustic Beacon

In this section we examine the performance of the algorithm given in Figure 2-4. Figures 4-3 through 4-50 contain the results of the Monte Carlo simulations conducted for scenarios 1 through 3. For each scenario, the initial value of the estimated parameter vector θ^* is taken as:

$$\theta_0^* = [\ 0 \quad 0 \quad -2500\ ]^T \tag{4-6}$$

The feasible region is defined by $Range_{max} = Depth_{max} = 5000$ meters. If the maximum likelihood estimate found using the algorithm of chapter II is outside the feasible region, the corresponding entries are left blank. To find a solution for these cases, we need to start with a different initial parameter vector θ^*, or use more experiment vectors.

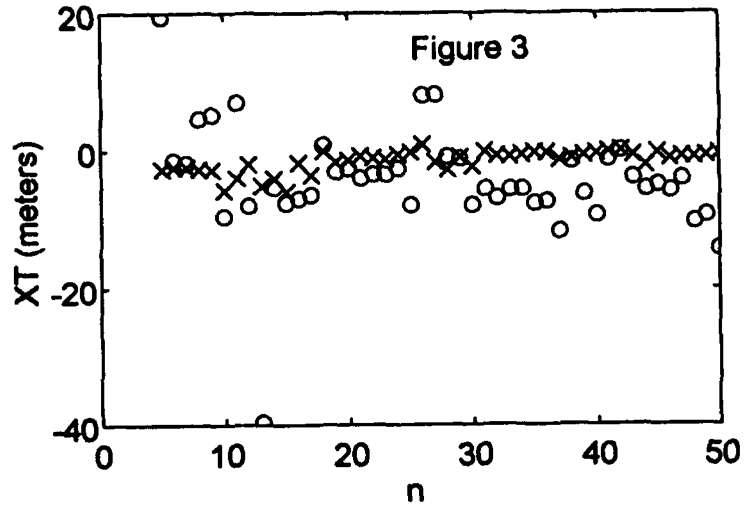
To test the effect of increasing the number of experiment vectors (n), we solve for the estimated parameter vector θ^* using n ranging from 5 to 50. We fix the effective sensor spacing at 45 pulse repetition intervals by using every 45th received beacon signal in defining the experiment vectors. To test the effects of increasing the effective sensor spacing, we solve for the estimated position vector using effective sensor spacings ranging from 2 to 90 pulse repetition intervals. For this case we fix the number of experiment vectors used to 25. We conduct each experiment using GPS errors of 3 meters rms and 25 meters rms. The results of the estimation algorithm for GPS errors of 3 meters rms are shown by x's on the plots, while the results of the

estimation algorithm for GPS errors of 25 meters rms are shown by o's.

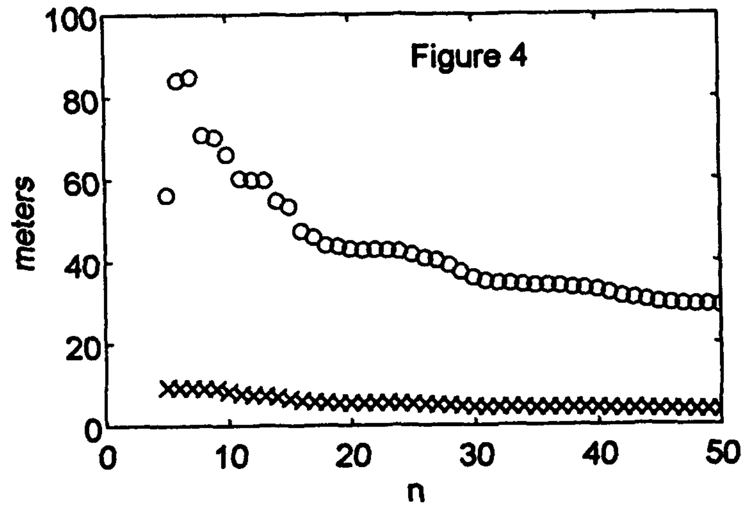
4.3.1 Scenario 1 - Stationary Acoustic Beacon Located at $\theta = [0\ 0\ -2000]^T$

Figures 4-3 through 4-10 show the results of the maximum likelihood estimation algorithm for different numbers of experiment vectors, and figures 4-11 through 4-18 show the results of the maximum likelihood estimation algorithm for different effective sensor spacings for scenario 1. We see that in general, as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated beacon position improves, and the standard deviation of the estimated position decreases. Also, as the effective sensor spacing increases, the accuracy of the estimated beacon position improves, and the standard deviation of the estimated position decreases. For both cases, the position errors and the standard deviation of the estimates initially decrease rapidly as the receiving ship spans more of its two dimensional search path. However after the number of experiment vectors used reaches approximately 20, or after the effective sensor spacing reaches approximately 35, the errors and the standard deviations decrease at much slower rate. At this point the receiving ship has completed two legs of the search path, and spans enough of the two dimensional search path to form a decent estimate of the acoustic beacon's position.

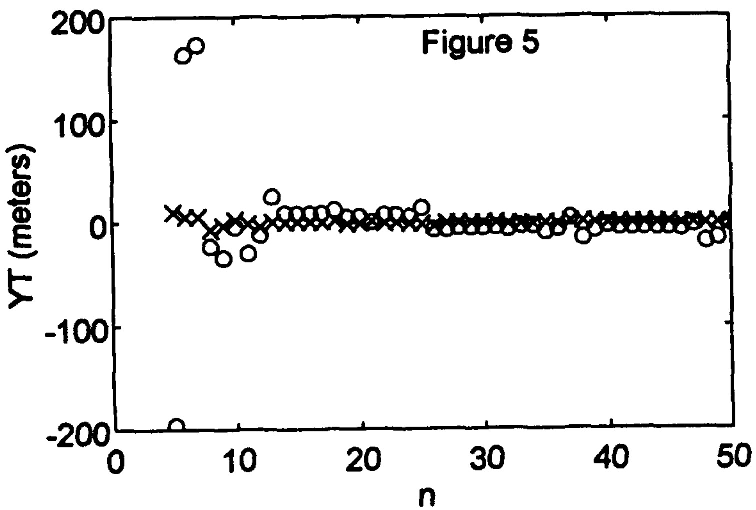
XT vs Number of Experiment Vectors



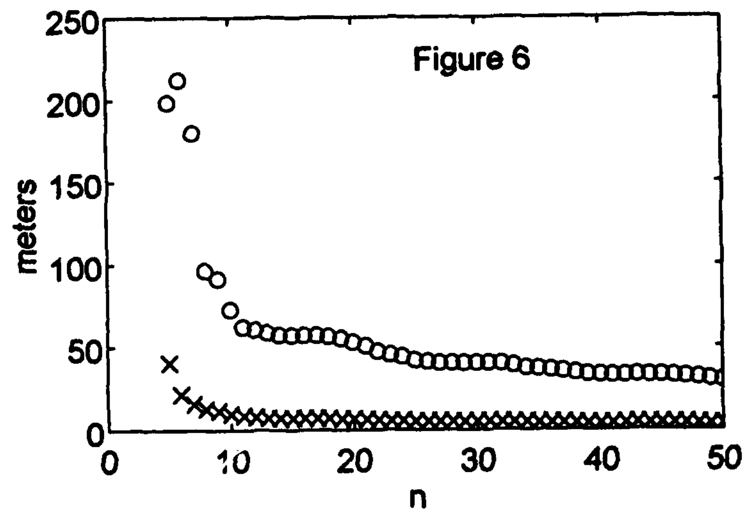
Standard Deviation of XT

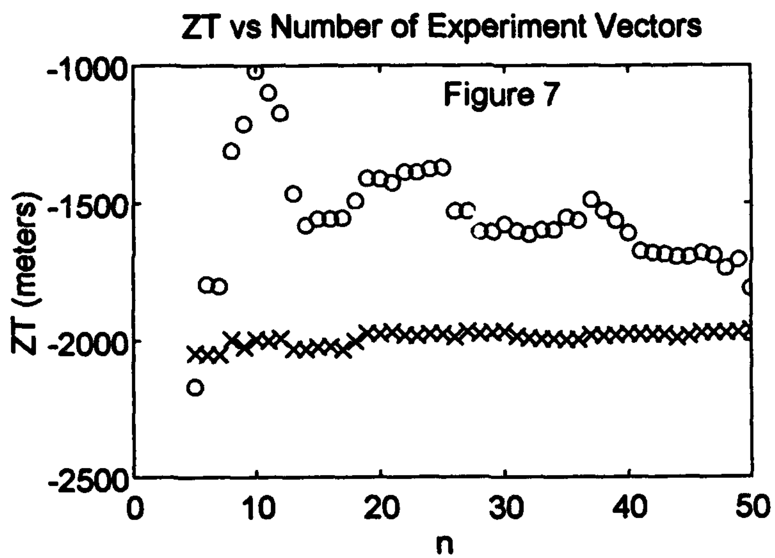


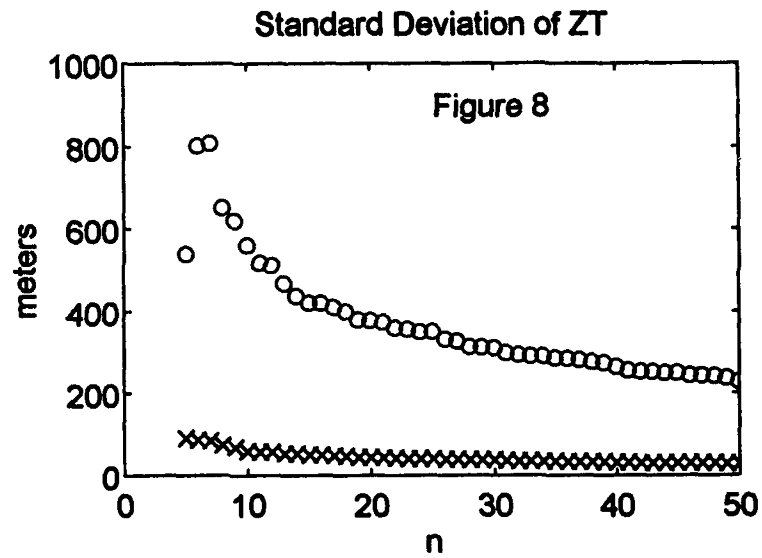
YT vs Number of Experiment Vectors

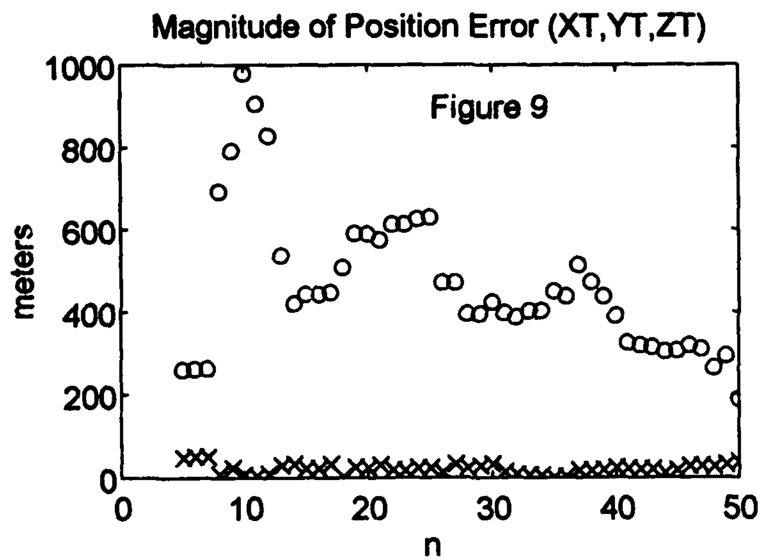


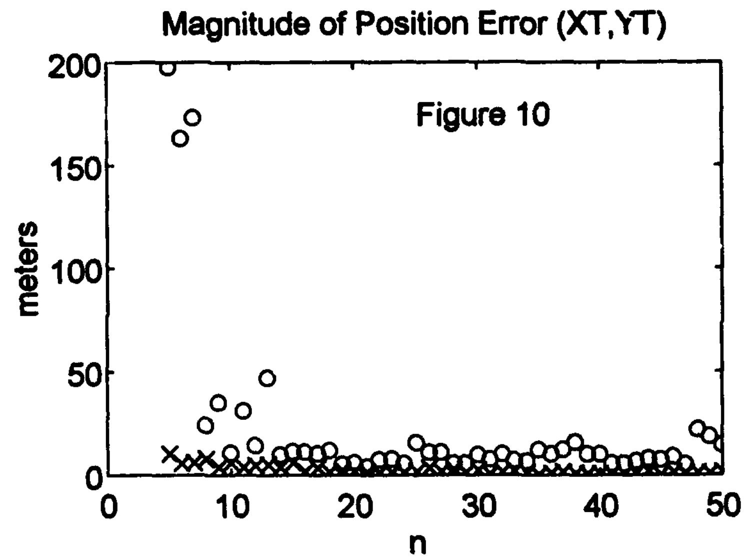
Standard Deviation of YT



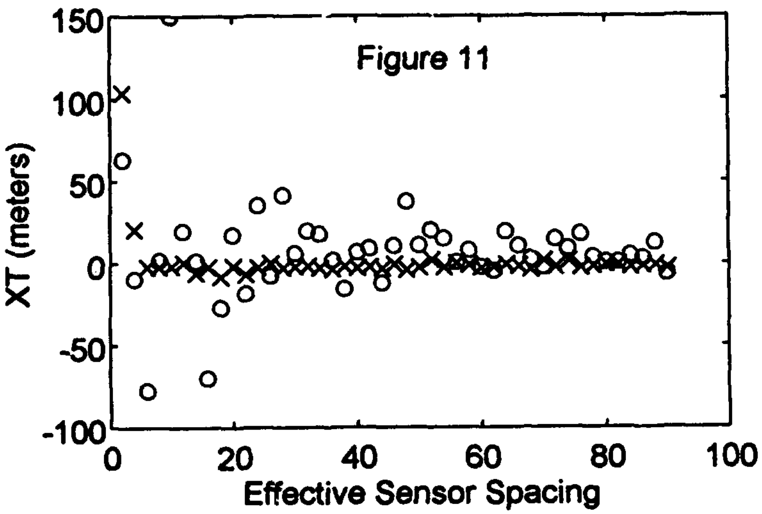

ZT vs Number of Experiment Vectors
Figure 7
ZT (meters)
-1000
-1500
-2000
-2500
0
10
20
30
40
50
n


Standard Deviation of ZT
Figure 8
meters
1000
800
600
400
200
0
10
20
30
40
50
n

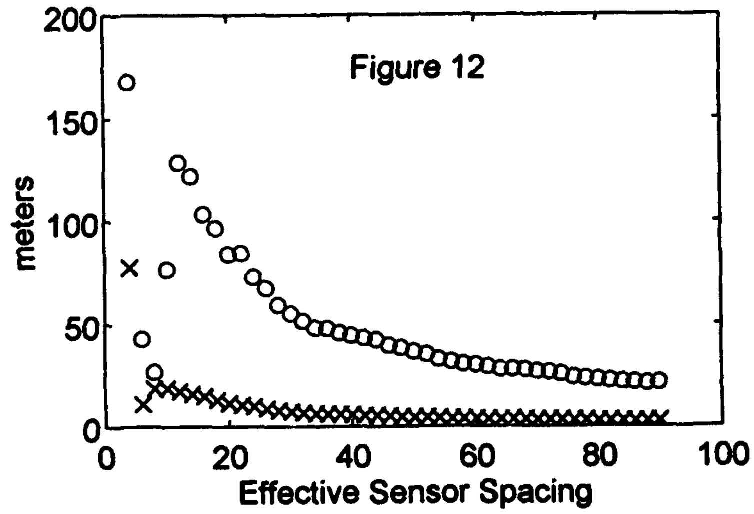

Magnitude of Position Error (XT,YT,ZT)
Figure 9
meters
1000
800
600
400
200
0
10
20
30
40
50
n


Magnitude of Position Error (XT,YT)
Figure 10
meters
200
150
100
50
0
10
20
30
40
50
n

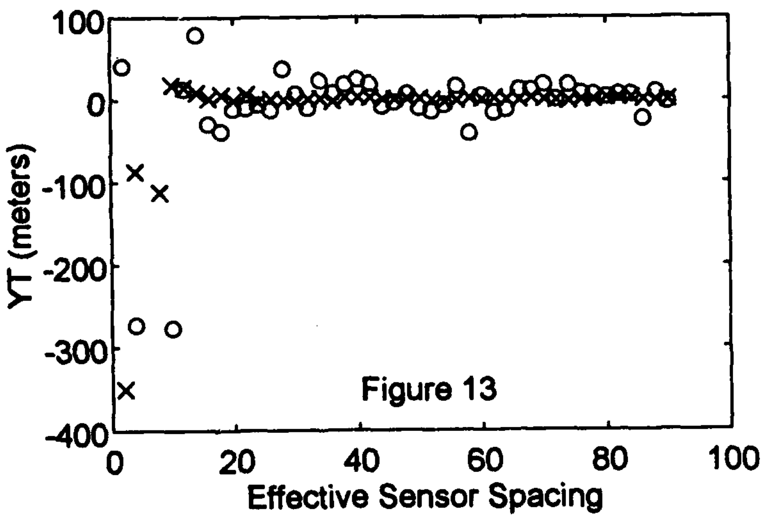
XT vs Effective Sensor Spacing



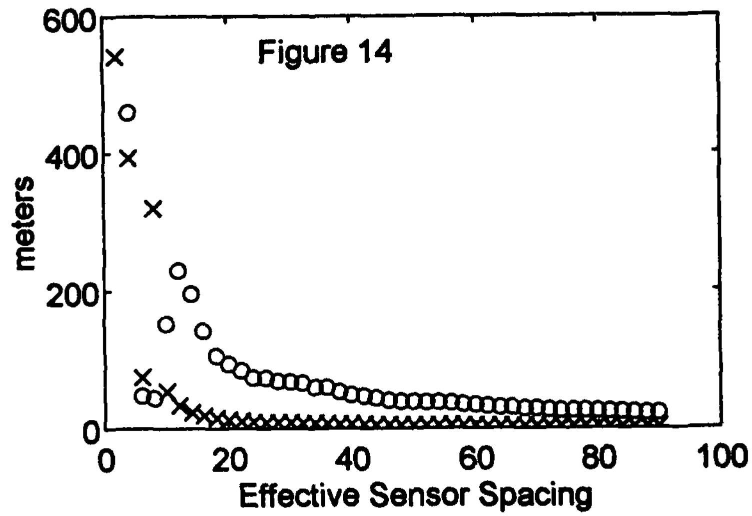
Standard Deviation of XT



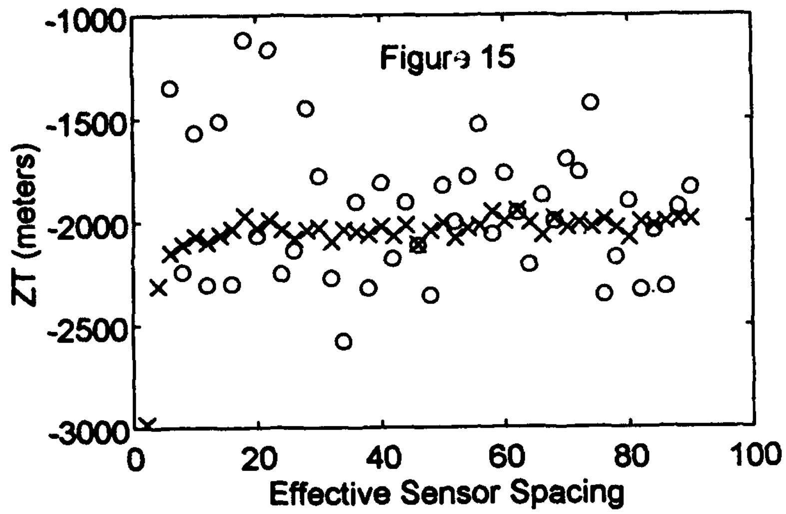
YT vs Effective Sensor Spacing



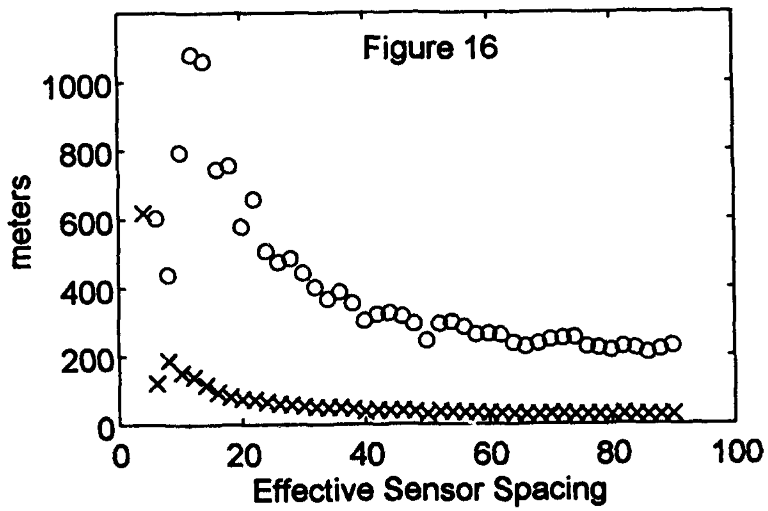
Standard Deviation of YT



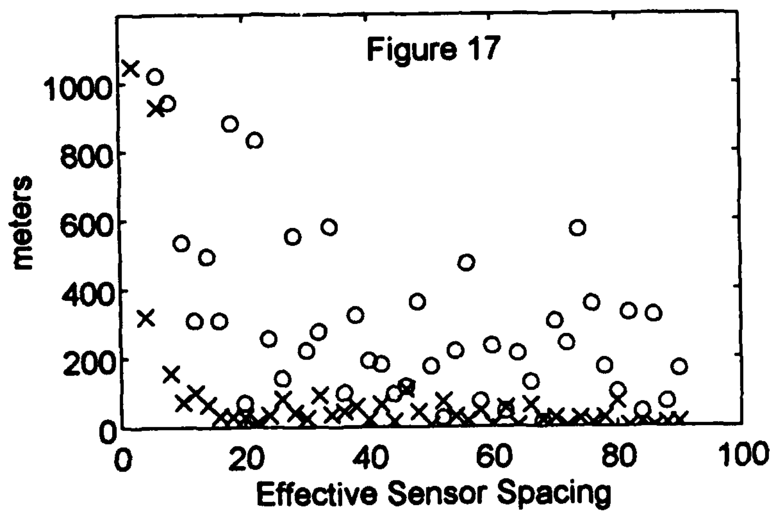
ZT vs Effective Sensor Spacing



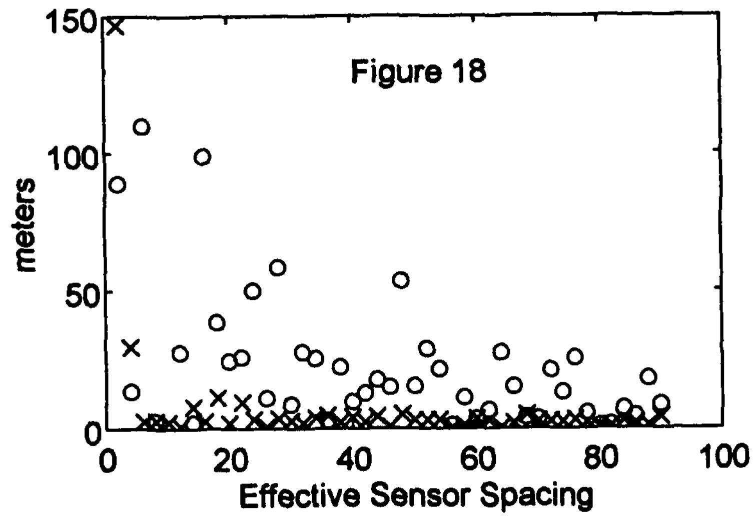
Standard Deviation of ZT



Magnitude of Position Error (XT,YT,ZT)



Magnitude of Position Error (XT,YT)

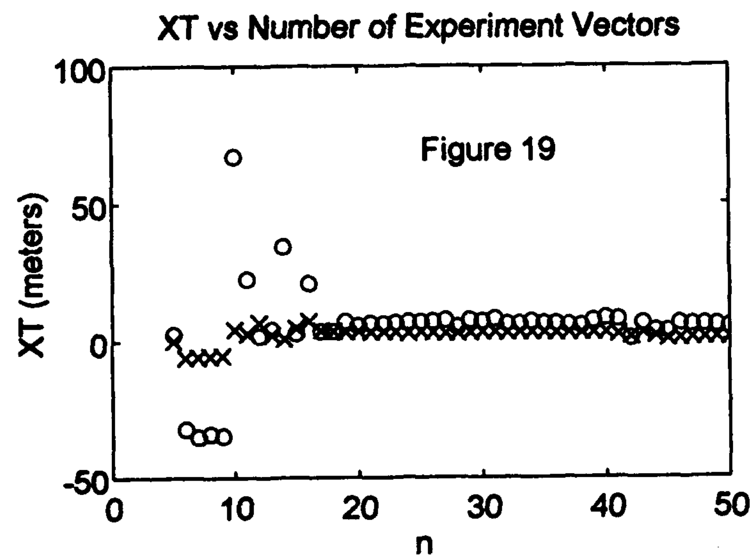


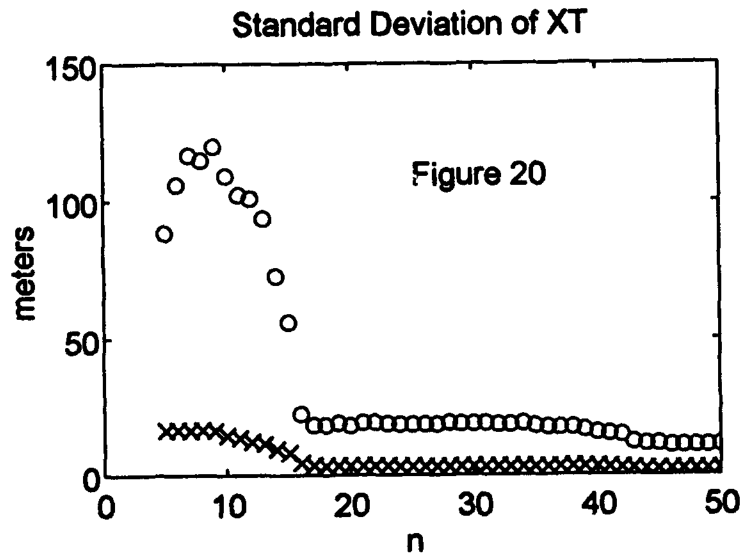
4.3.2 Scenario 2 - Stationary Acoustic Beacon Located at $\theta = [0\ 866\ -2000]^T$

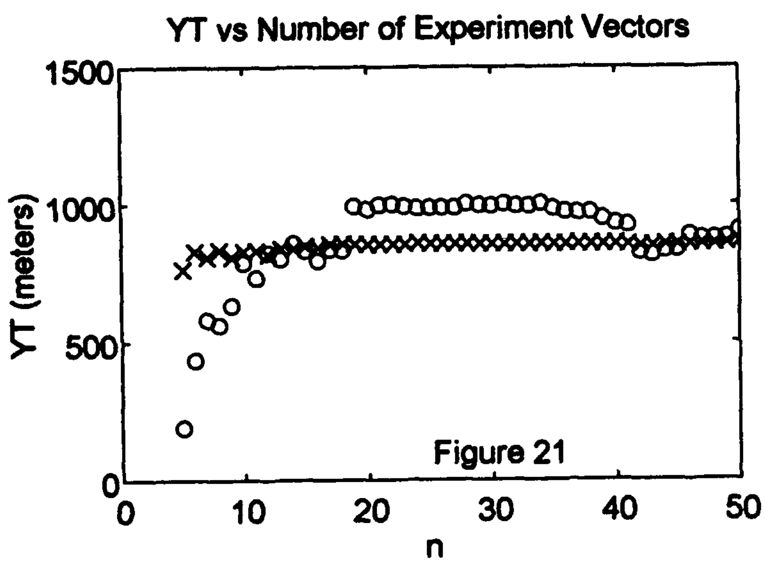
Figures 4-19 through 4-26 show the results of the maximum likelihood estimation algorithm for different numbers of experiment vectors, and figures 4-27 through 4-34 show the results of the maximum likelihood estimation algorithm for different effective sensor spacings for scenario 2. Like scenario 1, as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated beacon position improves, and the standard deviation of the estimated position decreases. Also like scenario 1, as the effective sensor spacing increases, the accuracy of the estimated beacon position improves, and the standard deviation of the estimated position decreases. Again we see an initial rapid decrease in the errors and standard deviations which levels off after the number of experiment vectors used reaches approximately 20, or the effective sensor spacing reaches approximately 35. As in scenario 1, these values correspond to when the receiving ship completes the second leg of the search path.

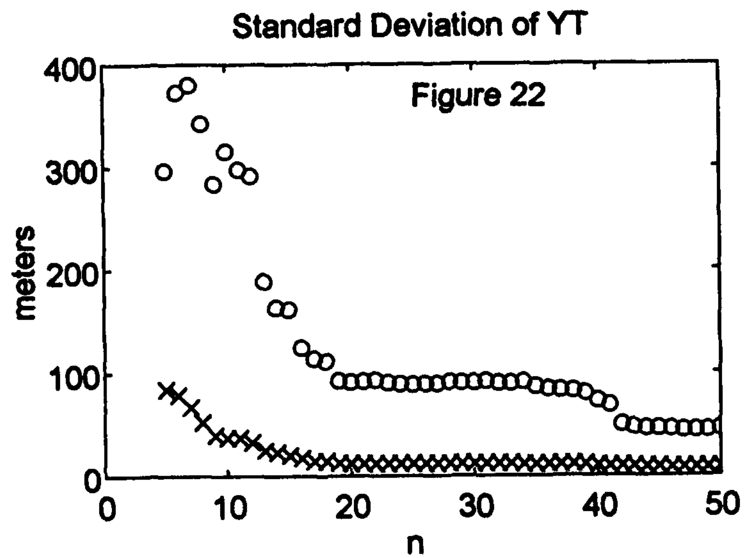
Unlike scenario 1, we see a significant improvement in the estimated position (especially when using 25 meter GPS errors) when the number of experiment vectors used is greater then 40, or the effective sensor spacing is greater then 65. At this point the receiving ship passes directly over the acoustic beacon and begins moving away from the beacon on the fourth leg of the search path. To see why we get this improvement, we can define a vertical plane as the plane orthogonal to the x-y plane that passes through the acoustic beacon's position and the starting point of the leg of the search path that the receiving ship is currently on. In general, it was found that if

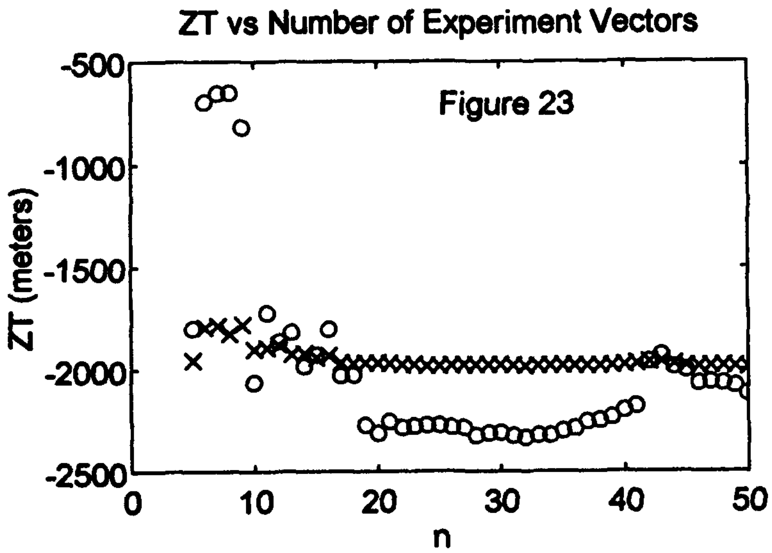
the receiving ship moves such that it stays in this vertical plane, significant improvements to the estimated beacon location can be made. This is because as the receiving ship moves in the same vertical plane relative to the acoustic beacon, the localization problem is essentially a two dimensional problem, and the range differences between measurement vectors is maximized for a given ship's velocity. The same is true if the acoustic beacon and the receiving ship are in the same horizontal plane.

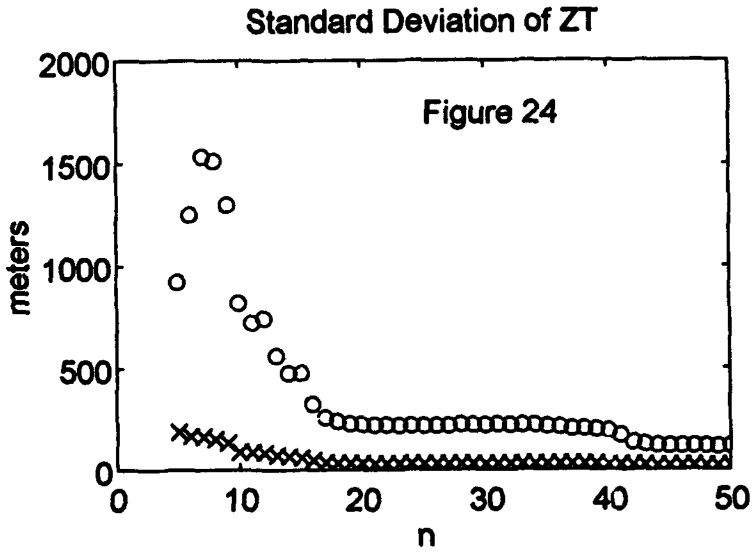

XT vs Number of Experiment Vectors
Figure 19
XT (meters)
100
50
0
-50
0
10
20
30
40
50
n

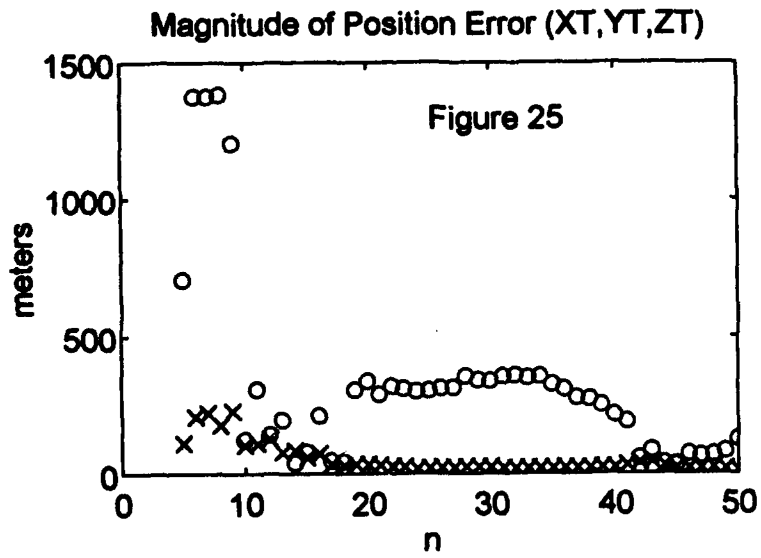

Standard Deviation of XT
Figure 20
meters
150
100
50
0
0
10
20
30
40
50
n

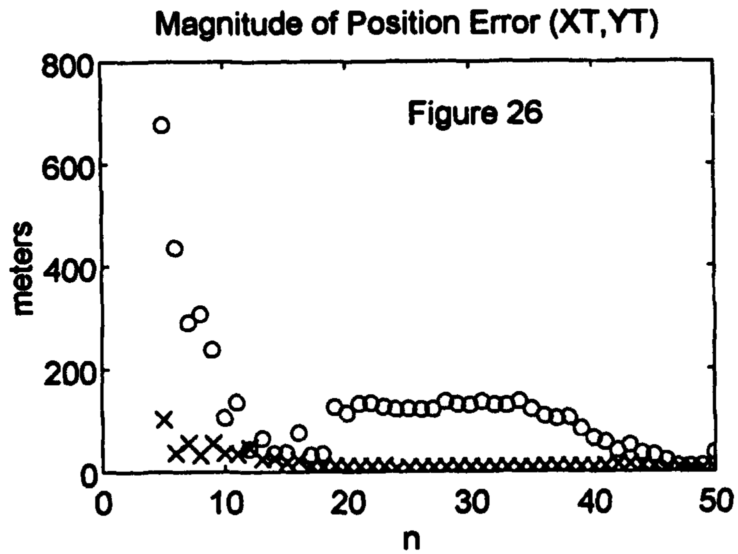

YT vs Number of Experiment Vectors
Figure 21
YT (meters)
1500
1000
500
0
0
10
20
30
40
50
n

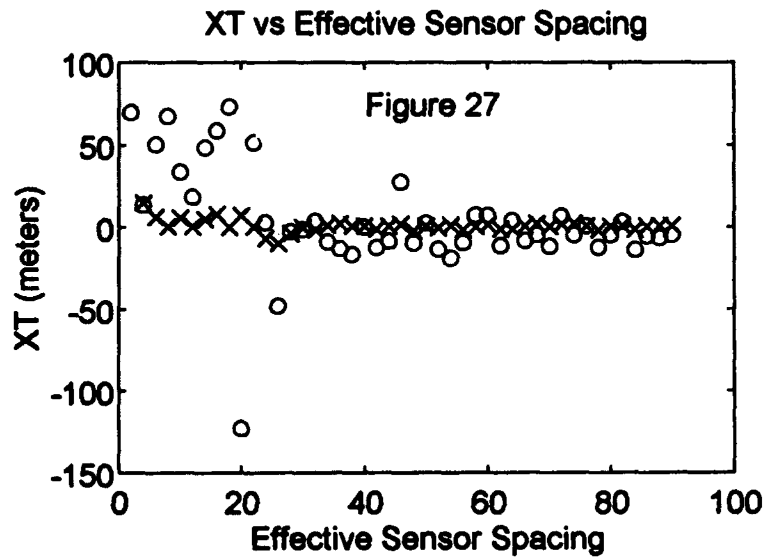

Standard Deviation of YT
Figure 22
meters
400
300
200
100
0
0
10
20
30
40
50
n

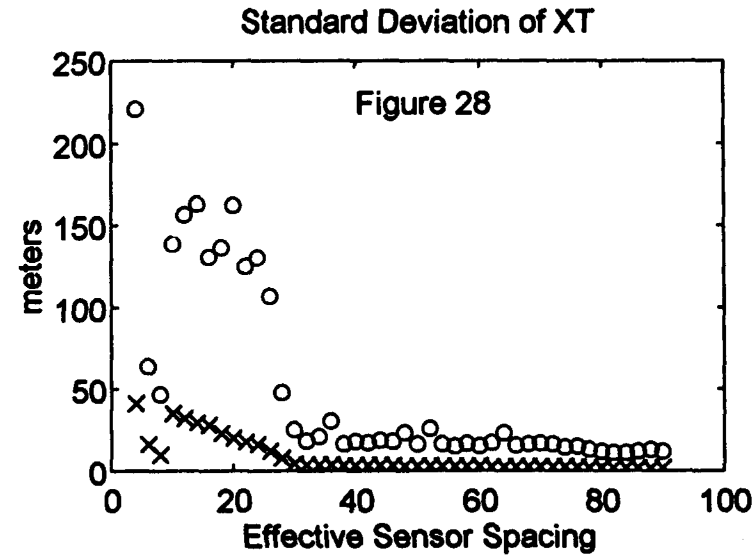

ZT vs Number of Experiment Vectors
Figure 23
ZT (meters)
-500
-1000
-1500
-2000
-2500
0
10
20
30
40
50
n

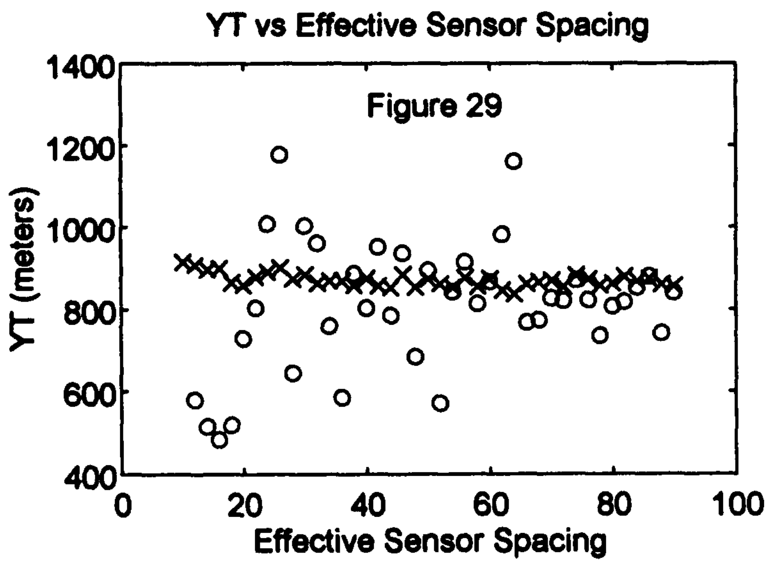

Standard Deviation of ZT
Figure 24
meters
2000
1500
1000
500
0
0
10
20
30
40
50
n

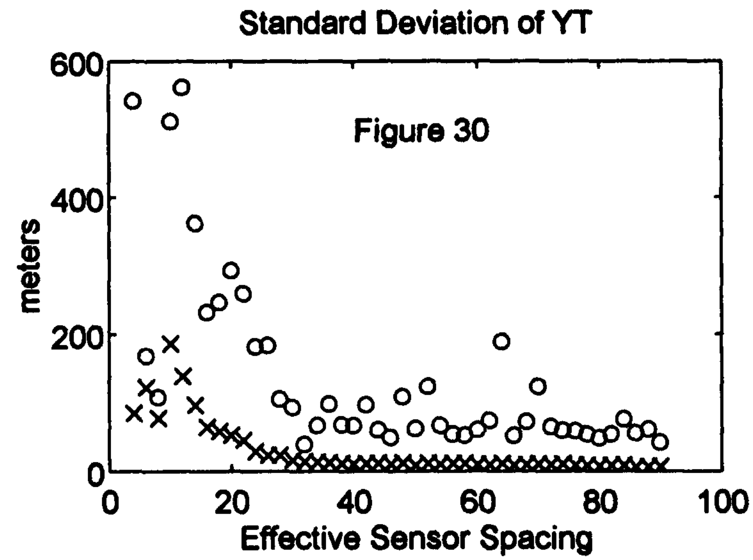

Magnitude of Position Error (XT,YT,ZT)
Figure 25
meters
1500
1000
500
0
0
10
20
30
40
50
n

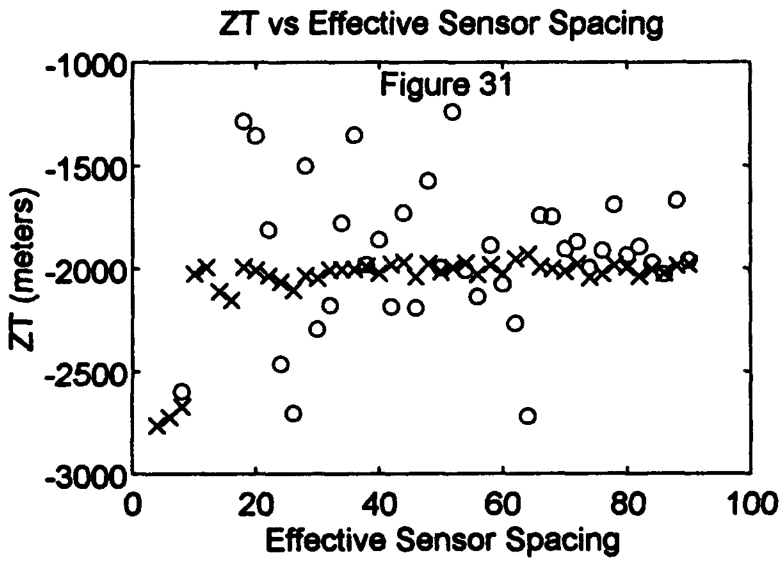

Magnitude of Position Error (XT,YT)
Figure 26
meters
800
600
400
200
0
0
10
20
30
40
50
n

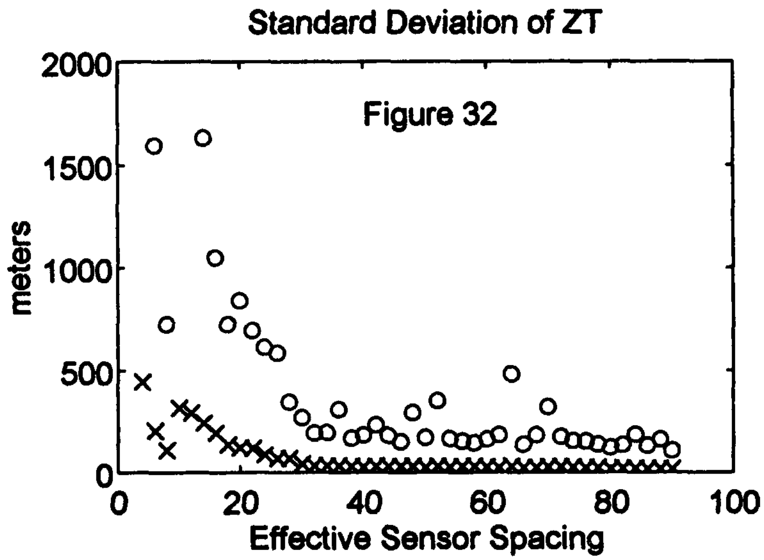

XT vs Effective Sensor Spacing
Figure 27
XT (meters)
100
50
0
-50
-100
-150
0
20
40
60
80
100
Effective Sensor Spacing

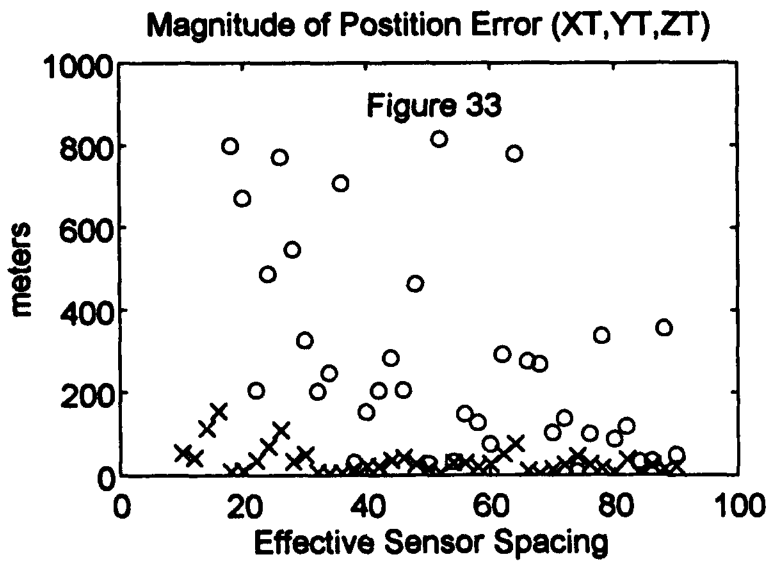

Standard Deviation of XT
Figure 28
meters
250
200
150
100
50
0
0
20
40
60
80
100
Effective Sensor Spacing

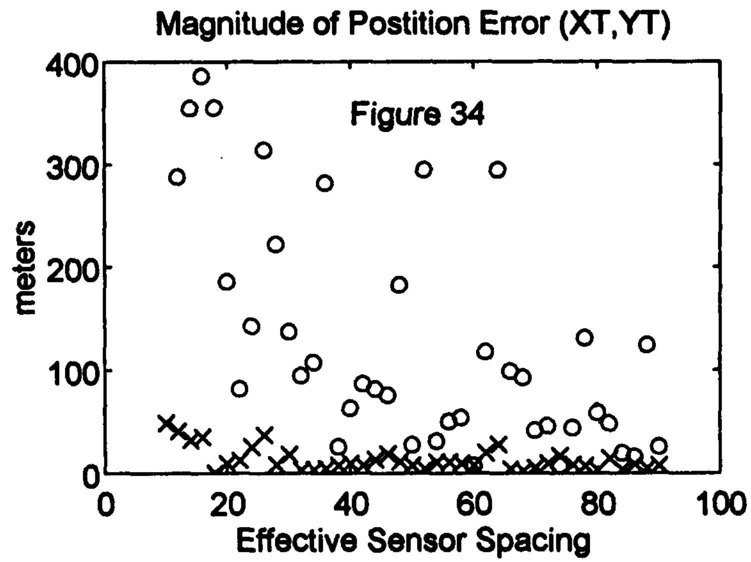

YT vs Effective Sensor Spacing
Figure 29
YT (meters)
1400
1200
1000
800
600
400
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of YT
Figure 30
meters
600
400
200
0
0
20
40
60
80
100
Effective Sensor Spacing


ZT vs Effective Sensor Spacing
Figure 31
ZT (meters)
-1000
-1500
-2000
-2500
-3000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of ZT
Figure 32
meters
2000
1500
1000
500
0
0
20
40
60
80
100
Effective Sensor Spacing

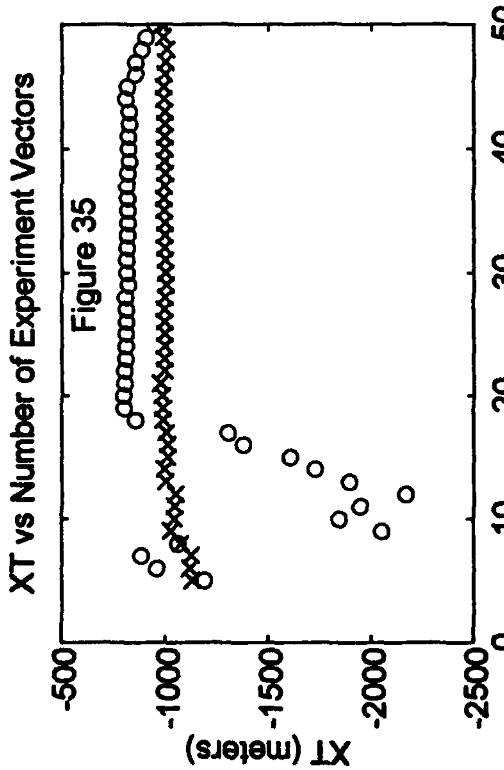

Magnitude of Postition Error (XT,YT,ZT)
Figure 33
meters
1000
800
600
400
200
0
0
20
40
60
80
100
Effective Sensor Spacing

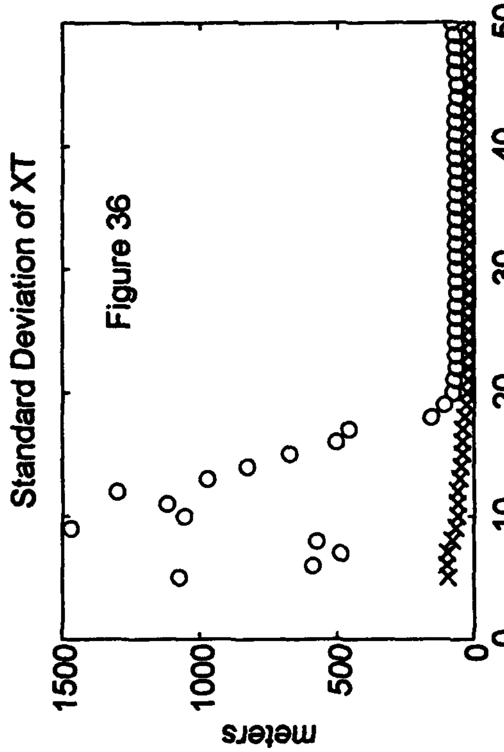

Magnitude of Postition Error (XT,YT)
Figure 34
meters
400
300
200
100
0
0
20
40
60
80
100
Effective Sensor Spacing

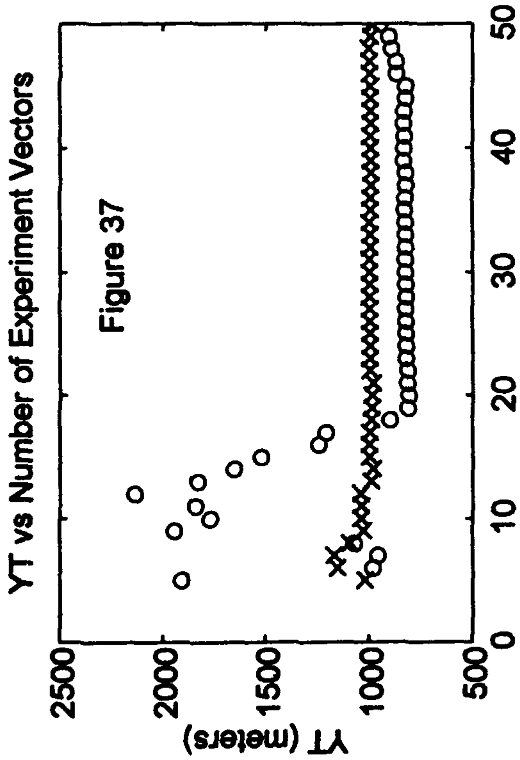
4.3.3 Scenario 3 - Stationary Acoustic Beacon Located at $\theta = [-1000\ 1000\ -2000]^T$

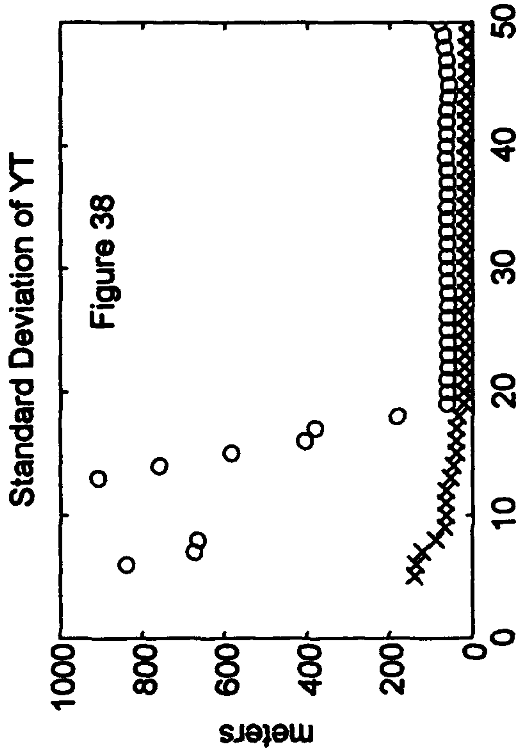
Figures 4-35 through 4-42 show the results of the maximum likelihood estimation algorithm for different numbers of experiment vectors, and figures 4-43 through 4-50 show the results of the maximum likelihood estimation algorithm for different effective sensor spacing for scenario 3. Like the first two scenarios, as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated beacon position improves, while the standard deviation of the estimated position decreases. Also, as the effective sensor spacing increases, the accuracy of the estimated beacon position improves, while the standard deviation of the estimated position decreases. Again we see an initial rapid decrease in the errors and standard deviations which levels off after the number of experiment vectors used reaches approximately 20, and the effective sensor spacing reaches approximately 35. As in the first two scenarios, these values correspond to when the receiving ship completes the second leg of the search path.

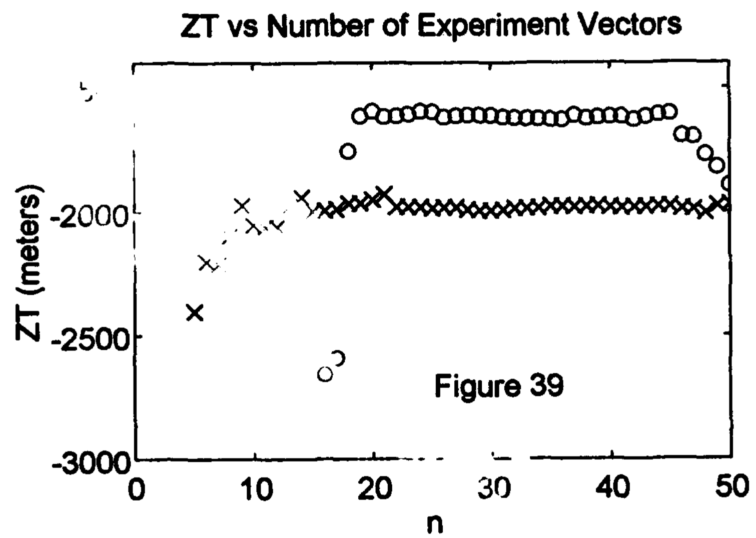
In this scenario, we see a significant improvement in the location estimate when the number of experiment vectors used is greater then 45, or when the effective element spacing is greater then 75. At this point the receiving ship passes through the closest point of approach to the acoustic beacon on the fifth leg of the search path. In general, it was found that the estimate of the acoustic beacon's location improves as the receiving ship passes through closest points of approach to the acoustic beacon along the search path.

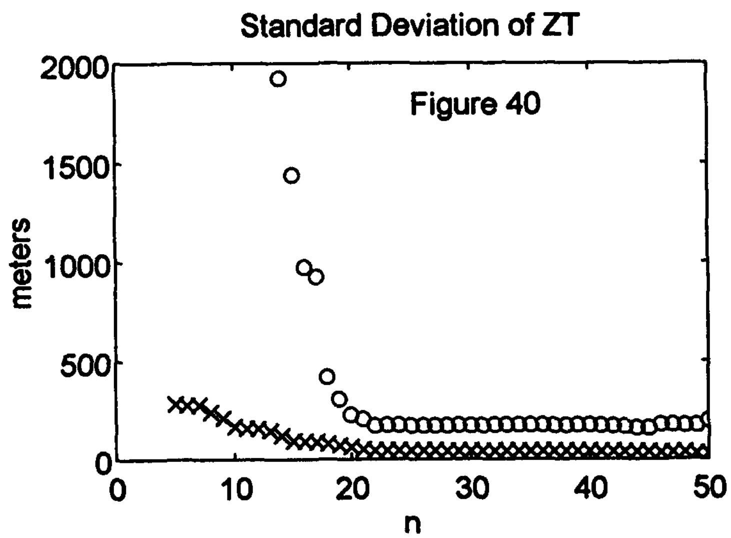

XT vs Number of Experiment Vectors
Figure 35
XT (meters)
n

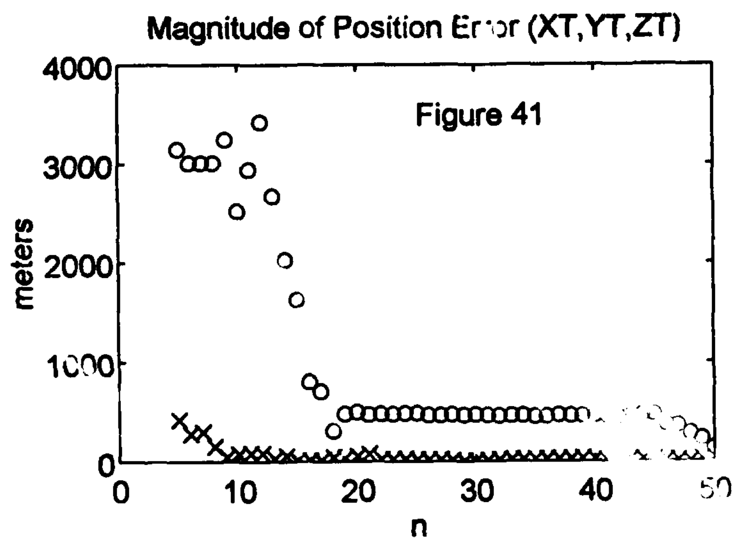

Standard Deviation of XT
Figure 36
meters
n

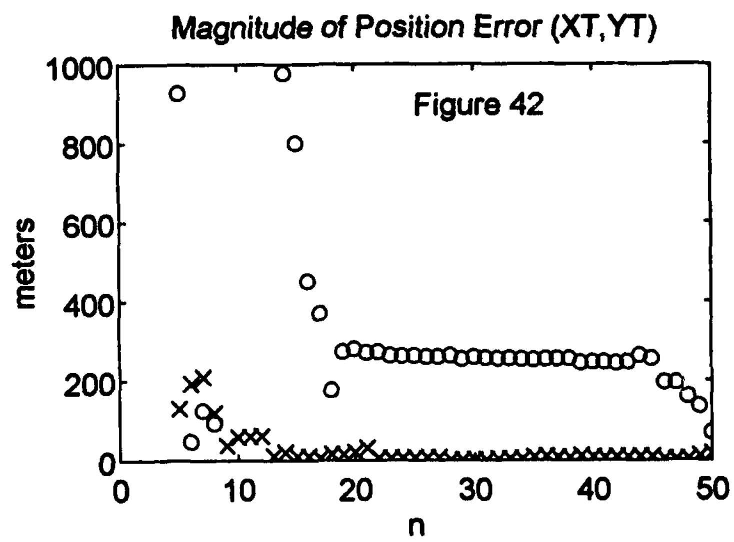

YT vs Number of Experiment Vectors
Figure 37
YT (meters)
n

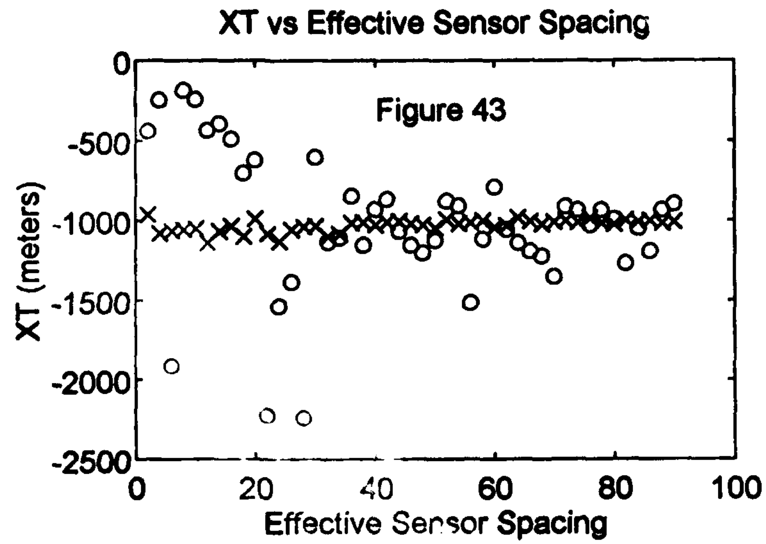

Standard Deviation of YT
Figure 38
meters
n

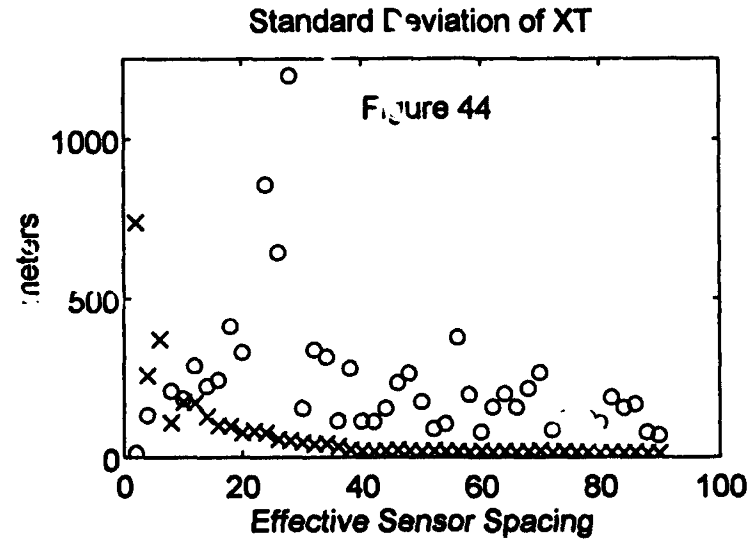

ZT vs Number of Experiment Vectors
ZT (meters)
-2000
-2500
-3000
0
10
20
30
40
50
n
Figure 39

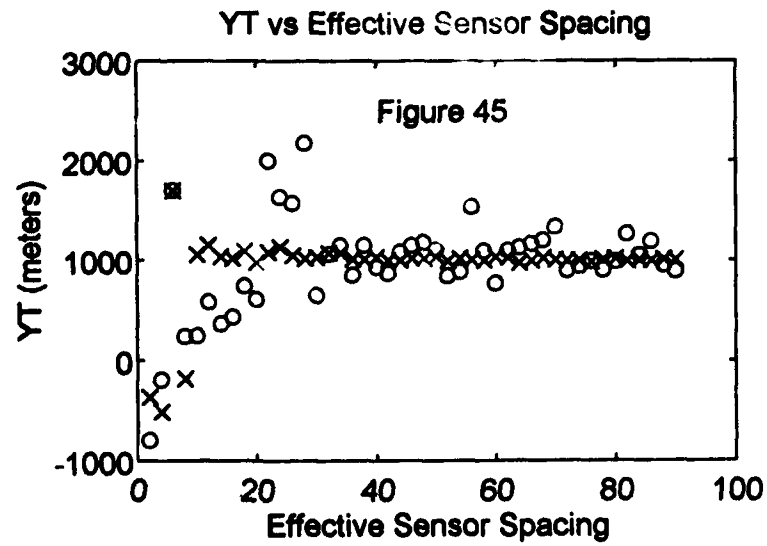

Standard Deviation of ZT
meters
2000
1500
1000
500
0
0
10
20
30
40
50
n
Figure 40

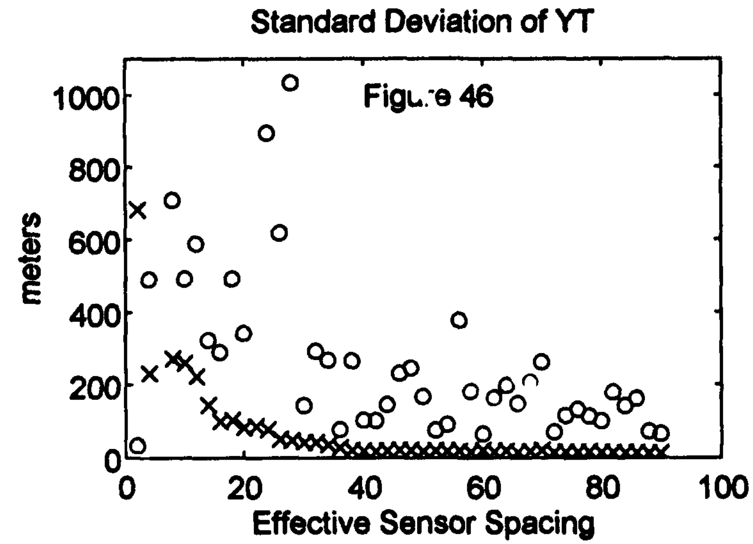

Magnitude of Position Error (XT,YT,ZT)
meters
4000
3000
2000
1000
0
0
10
20
30
40
50
n
Figure 41

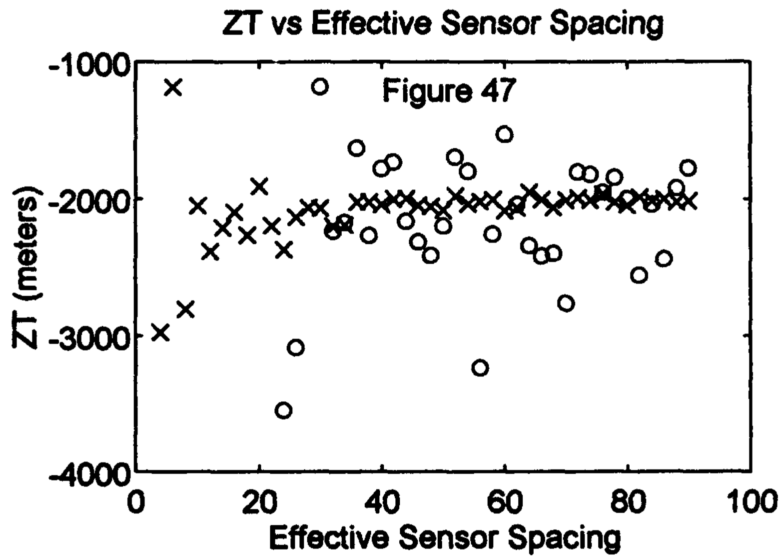

Magnitude of Position Error (XT,YT)
meters
1000
800
600
400
200
0
0
10
20
30
40
50
n
Figure 42

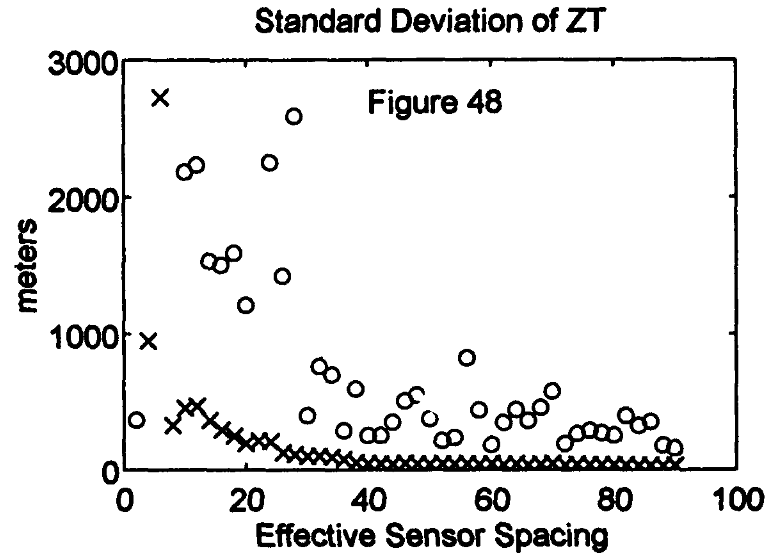

XT vs Effective Sensor Spacing
Figure 43
XT (meters)
Effective Sensor Spacing

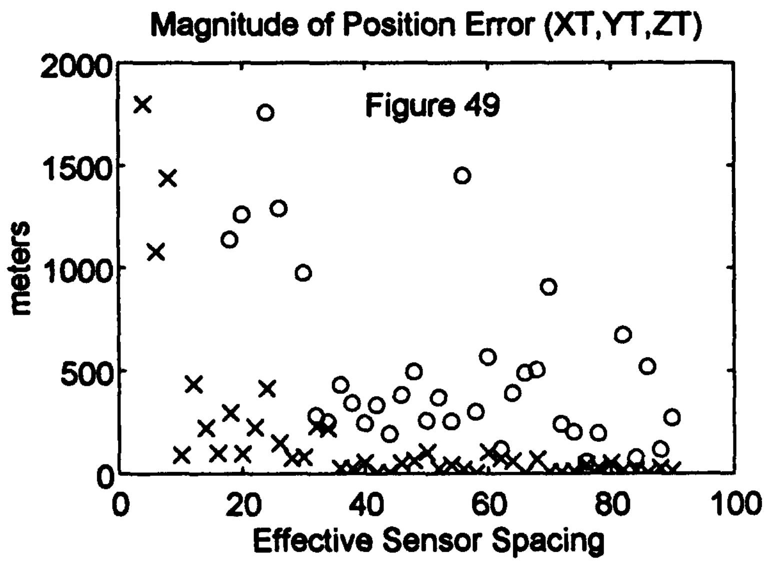

Standard Deviation of XT
Figure 44
meters
Effective Sensor Spacing

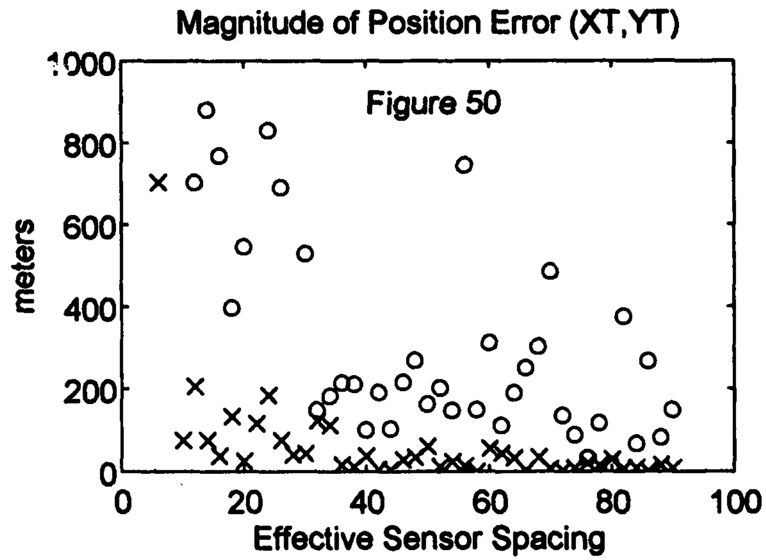

YT vs Effective Sensor Spacing
Figure 45
YT (meters)
Effective Sensor Spacing


Standard Deviation of YT
Figure 46
meters
Effective Sensor Spacing


ZT vs Effective Sensor Spacing
Figure 47
ZT (meters)
-1000
-2000
-3000
-4000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of ZT
Figure 48
meters
3000
2000
1000
0
20
40
60
80
100
Effective Sensor Spacing


Magnitude of Position Error (XT,YT,ZT)
Figure 49
meters
2000
1500
1000
500
0
20
40
60
80
100
Effective Sensor Spacing


Magnitude of Position Error (XT,YT)
Figure 50
meters
1000
800
600
400
200
0
20
40
60
80
100
Effective Sensor Spacing

4.3.4 Stationary Acoustic Beacon Summary

From scenarios 1 through 3 we showed that as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated beacon position improves, while the standard deviation of the estimated position decreases. Also, as the effective sensor spacing increases, the accuracy of the estimated beacon position improves, while the standard deviation of the estimated position decreases. We also showed that as the receiving ship passes through a closest point of approach, the estimated position of the acoustic beacon improves. This is why the estimate of acoustic beacon's location in scenario 1 is better then the estimates obtained for the other two scenarios. In scenario 1, each leg of the search path passes through a closest point of approach. This is not true for the other two scenarios. In all of the scenarios, we see that the depth of the acoustic beacon is the least accurately estimated parameter. This is because we are attempting to locate the acoustic beacon in three dimensions, while the receiving ship only moves in two dimensions. Finally, we see that the accuracy of the estimated acoustic beacon position obtained from using GPS errors of 3 meters is approximately an order of magnitude better then the estimated acoustic beacon position obtained from using GPS errors of 25 meters.

These observations suggest that to obtain the best possible estimate of the acoustic beacon's location, we need to use a large number of experiment vectors with a large effective sensor separation, and we need to use a Differential/P code GPS receiver. The problem with this is that the time it takes to collect the measurement vectors used to form the experiment vectors increases with increasing sensor spacing

and increasing numbers of experiment vectors. As an example, with an effective sensor spacing of 45 pulse repetition intervals, a pulse repetition interval of 2 seconds, and using 50 experiment vectors it takes over 76 minutes to collect the required measurement vectors. If time is an issue, then we must use fewer experiment vectors and shorter effective sensor spacings. From looking at the results of the simulations presented above, for the hexagonal search path of figure 4-1 and for a pulse repetition interval of 2 seconds, the minimum required number of experiment vectors is 20, and the minimum effective sensor spacing is 35. With these values we span enough of the two dimensional search path to obtain a decent estimate of the acoustic beacon's location.

4.4 Moving Acoustic Beacon

In this section we examine the performance of the algorithm given in Figure 3-1. Figures 4-51 through 4-146 contain the results of the Monte Carlo simulations conducted for scenarios 4 through 6. For each scenario, the initial value of the estimated parameter vector θ^* is taken as:

$$\theta_o^* = [\ 0 \quad 0 \quad -2500 \quad 0 \quad 0 \quad 0\ ]^T \tag{4-7}$$

The feasible region is defined by $\text{Range}_{max} = \text{Depth}_{max} = 5000$ meters, and $v_{xmax} = v_{ymax} = v_{zmax} = 5$ meters per second. If the maximum likelihood estimate found using the algorithm of chapter III is outside the feasible region, the corresponding entries are left blank. To find a solution for these cases, we need to start with a different initial parameter vector θ, or use more experiment vectors.

To test the effect of increasing the number of experiment vectors (n), we solve

for the estimated parameter vector θ^* using n ranging from 10 to 50. We fix the effective sensor spacing at 60 pulse repetition intervals by using every 60th received beacon signal in defining the experiment vectors. To test the effect of increasing the effective sensor spacing, we solve for the estimated position vector using effective sensor spacings ranging from 2 to 90 pulse repetition intervals. For this case we fix the number of experiment vectors used to 30. We conduct each experiment using GPS errors of 3 meters rms and 25 meters rms. The results of the estimation algorithm for GPS errors of 3 meters rms are shown by x's on the plots, while the results of the estimation algorithm for GPS errors of 25 meters rms is shown by o's.

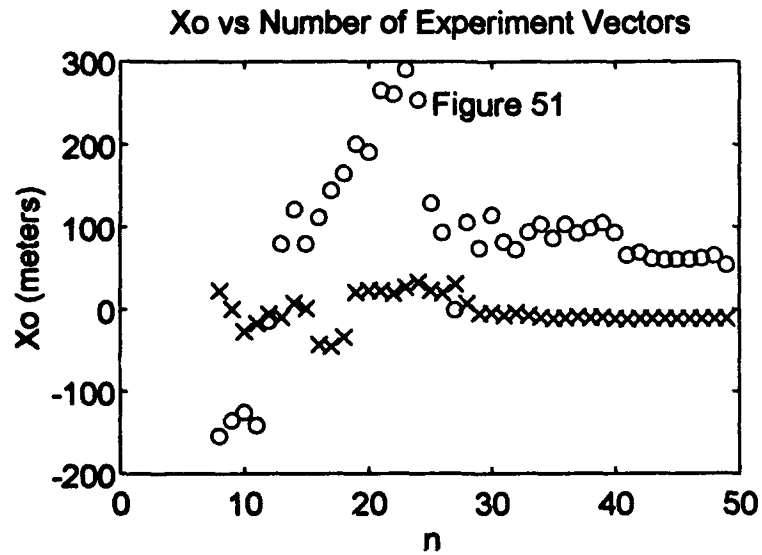
4.4.1 Scenario 4 - Moving Acoustic Beacon Located at:

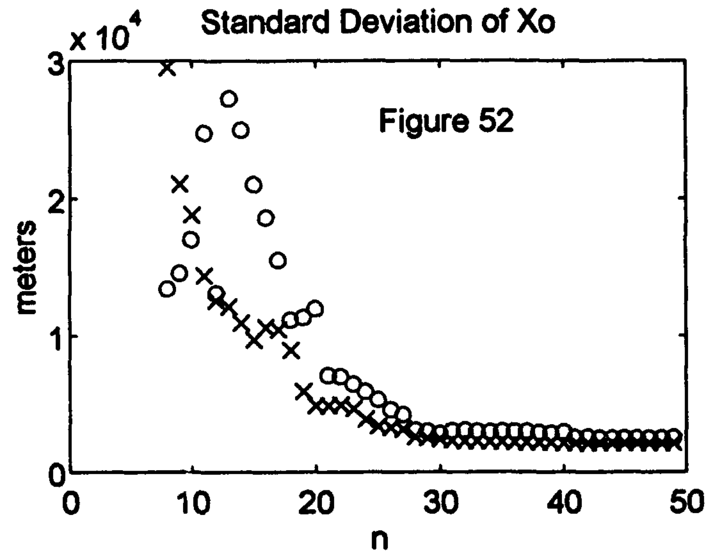
$$\theta = [0\ 0\ -2000\ 0.5\ -0.5\ 0.25]^T$$

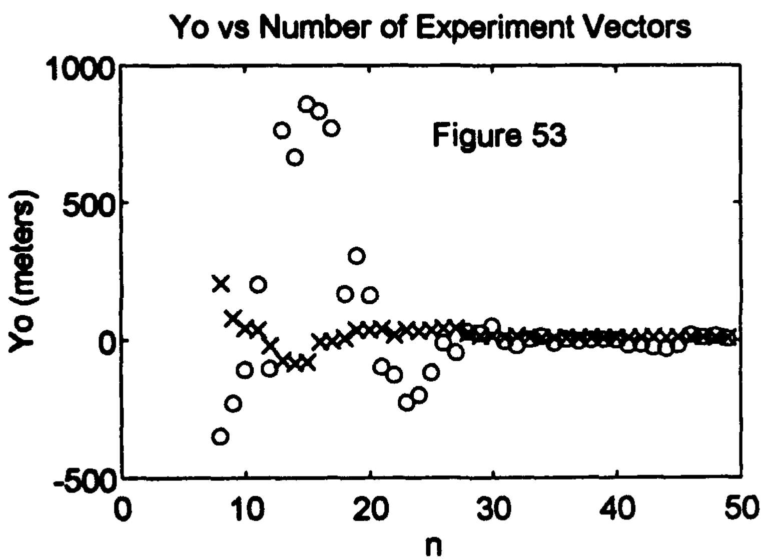
Figures 4-51 through 4-66 show the results of the maximum likelihood estimation algorithm for different numbers of experiment vectors, and figures 4-67 through 4-82 show the results of the maximum likelihood estimation algorithm for different effective sensor spacings for scenario 4. Similar to the stationary acoustic beacon scenarios, we see that as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, while the standard deviation of the estimated position and velocities decrease. We also see that as the effective sensor spacing increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, and the standard deviation of the estimated position decreases. As with the stationary acoustic beacon, the position and velocity errors

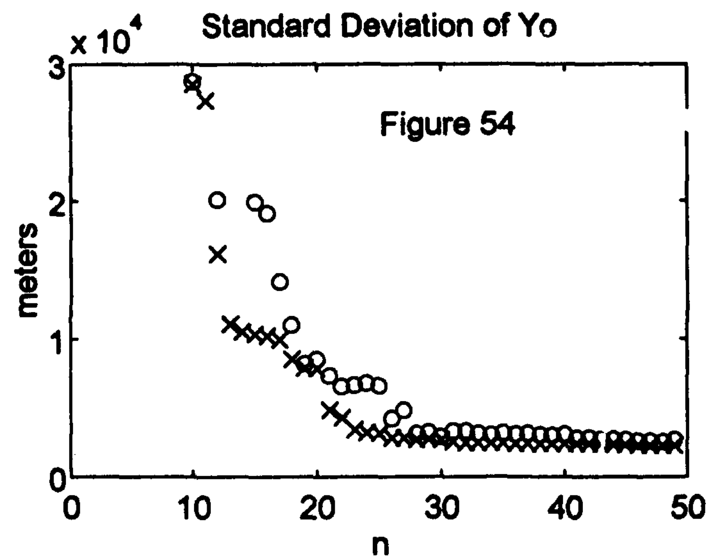
initially decrease rapidly, however, after the number of experiment vectors used reaches approximately 30, or the effective sensor spacing reaches approximately 45, the errors and the standard deviations decrease much slower.

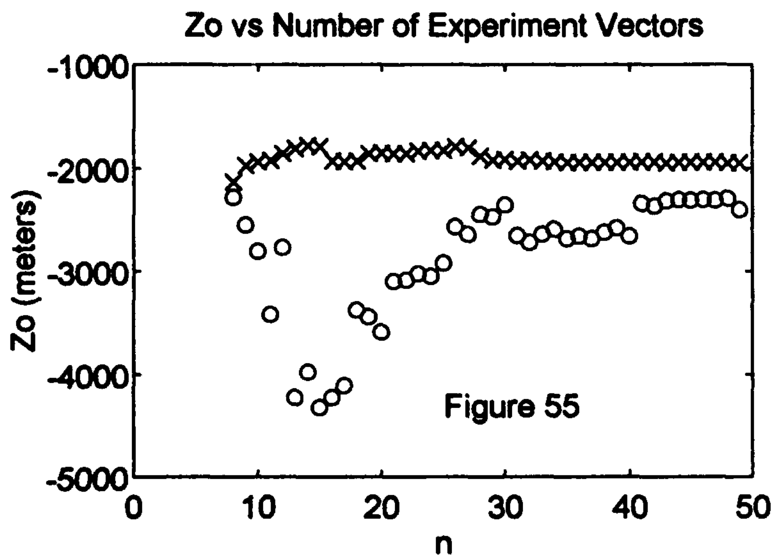
Figures 4-61 through 4-66 and figures 4-77 through 4-82 show the estimation algorithm's ability to track the linearly moving acoustic beacon. We see that projected current positions follow the linearly moving acoustic beacon fairly well when using 3 meter GPS errors. We also see that the standard deviation of the projected current positions decrease as the number of experiment vectors used increases and the effective sensor spacing increases. However, we notice in a few cases the standard deviations increase slightly for increasing numbers of experiment vectors used and increasing effective sensor spacing. This is a result of projecting the current position forward in time using equation 3-19.

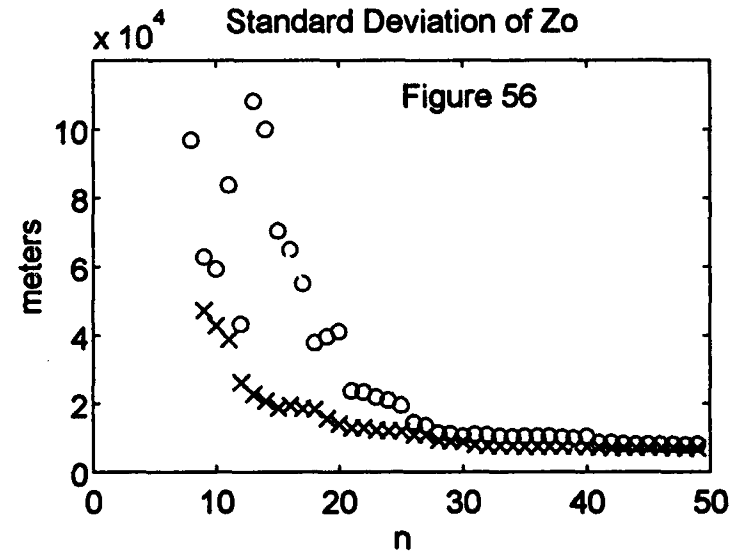

Xo vs Number of Experiment Vectors
Figure 51
Xo (meters)
300
200
100
0
-100
-200
0
10
20
30
40
50
n

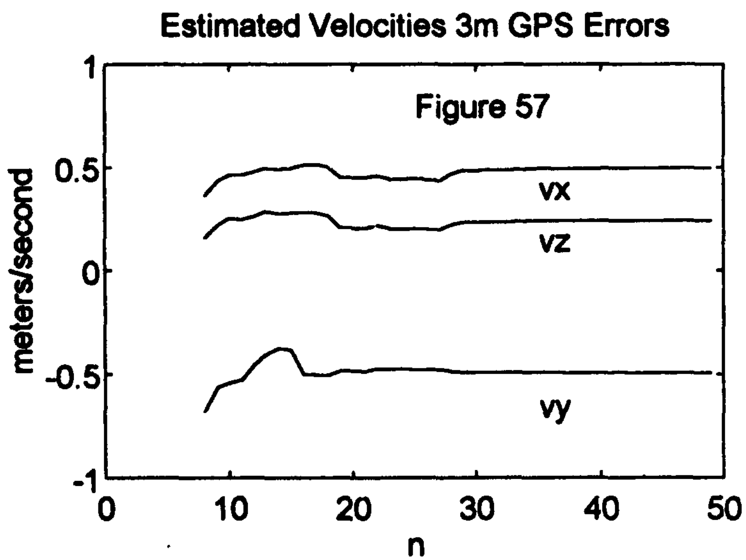

Standard Deviation of Xo
x 10^4
Figure 52
meters
3
2
1
0
0
10
20
30
40
50
n

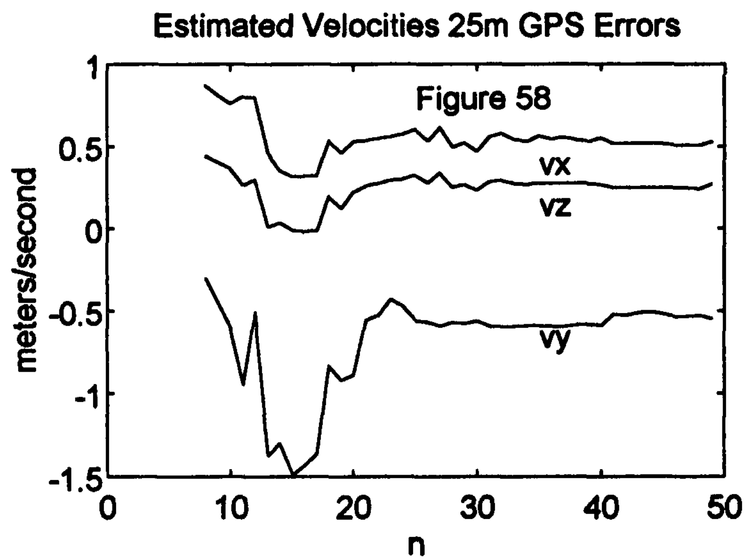

Yo vs Number of Experiment Vectors
Figure 53
Yo (meters)
1000
500
0
-500
0
10
20
30
40
50
n

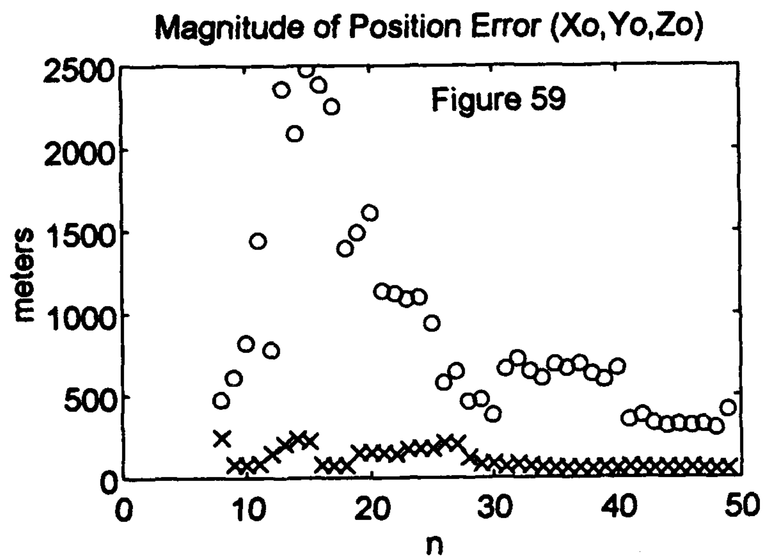

Standard Deviation of Yo
x 10^4
Figure 54
meters
3
2
1
0
0
10
20
30
40
50
n

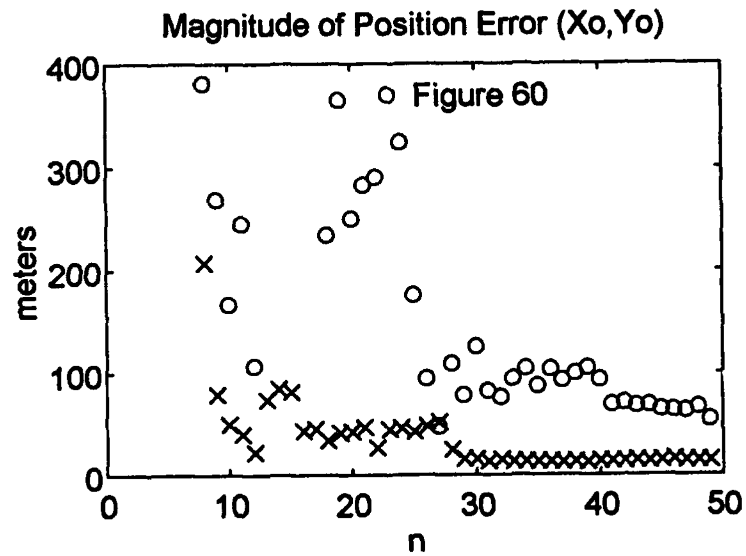

Zo vs Number of Experiment Vectors
Figure 55
Zo (meters)
-1000
-2000
-3000
-4000
-5000
0
10
20
30
40
50
n

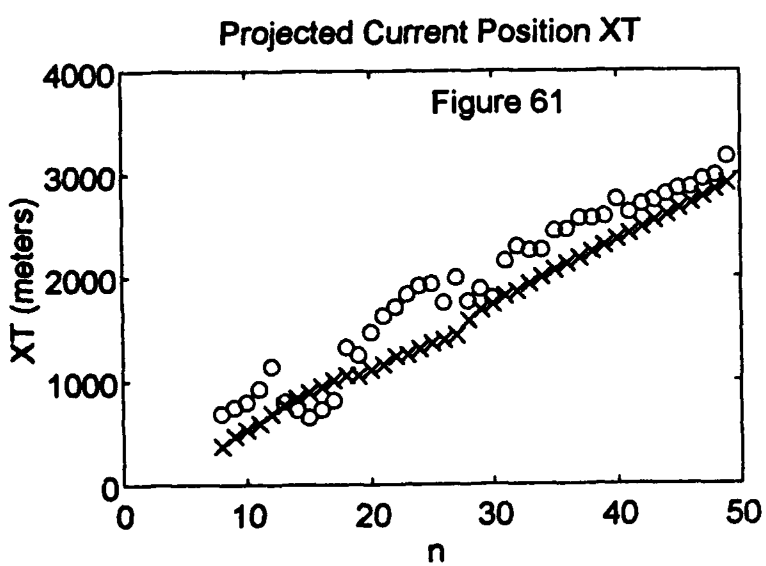

Standard Deviation of Zo
x 10^4
Figure 56
meters
10
8
6
4
2
0
0
10
20
30
40
50
n

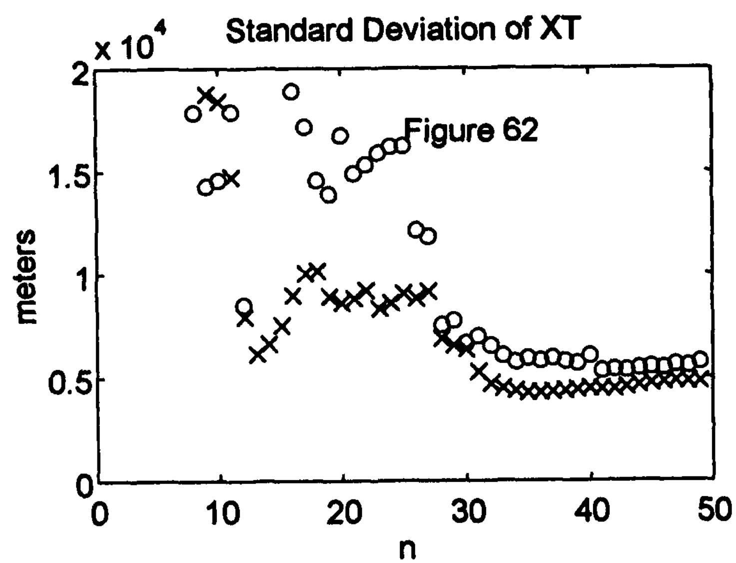

Estimated Velocities 3m GPS Errors
Figure 57
vx
vz
vy
meters/second
1
0.5
0
-0.5
-1
0
10
20
30
40
50
n

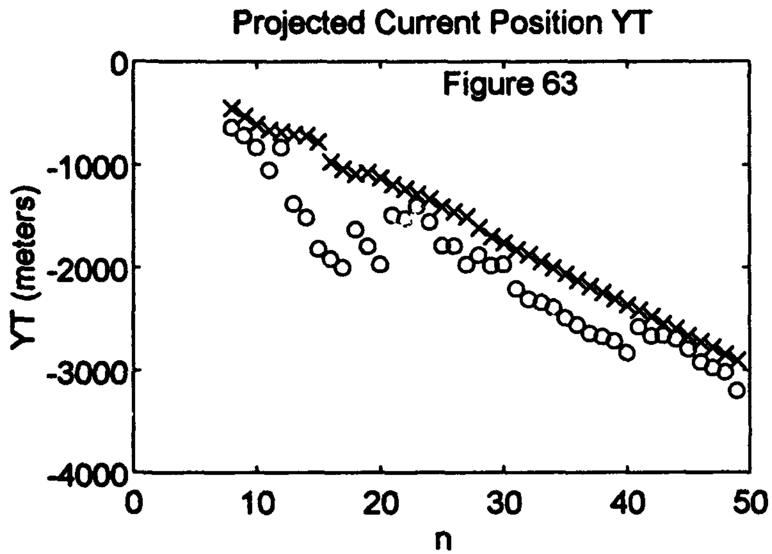

Estimated Velocities 25m GPS Errors
Figure 58
vx
vz
vy
meters/second
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
n

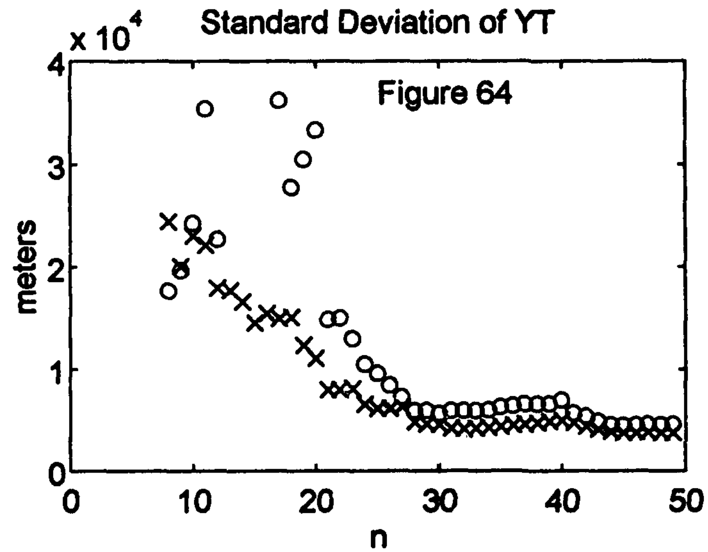

Magnitude of Position Error (Xo,Yo,Zo)
Figure 59
meters
n

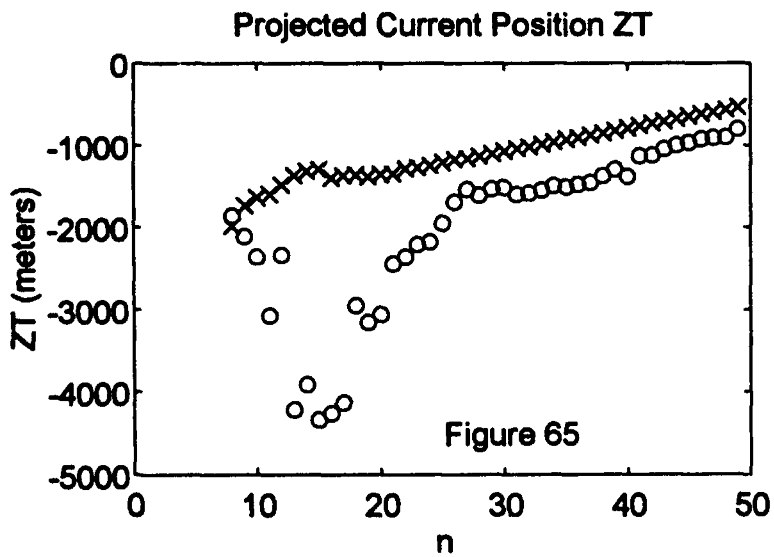

Magnitude of Position Error (Xo,Yo)
Figure 60
meters
n

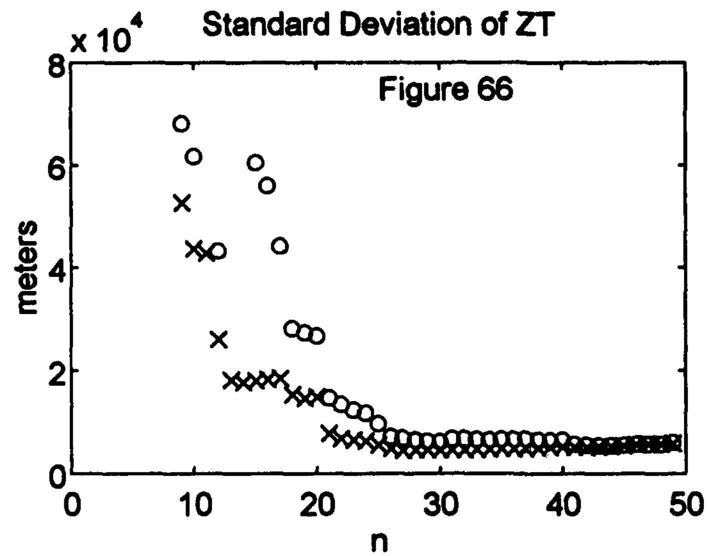

Projected Current Position XT
Figure 61
XT (meters)
n

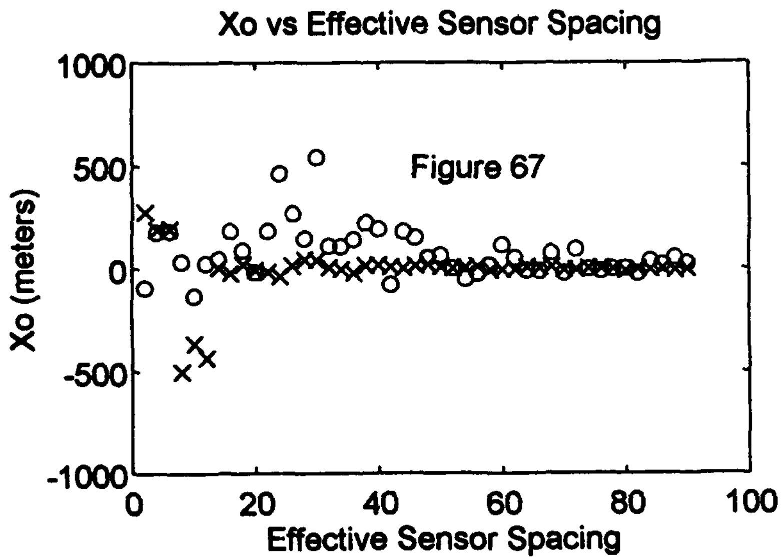

Standard Deviation of XT
Figure 62
x 10^4
meters
n

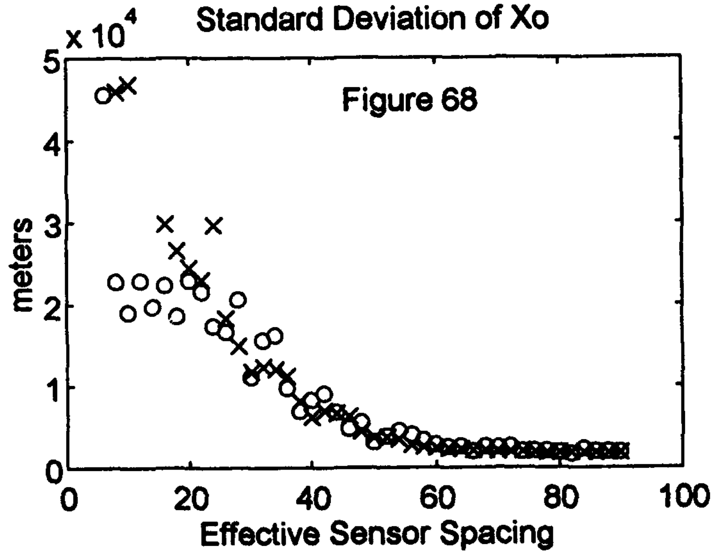

Projected Current Position YT
Figure 63
YT (meters)
0
-1000
-2000
-3000
-4000
0
10
20
30
40
50
n

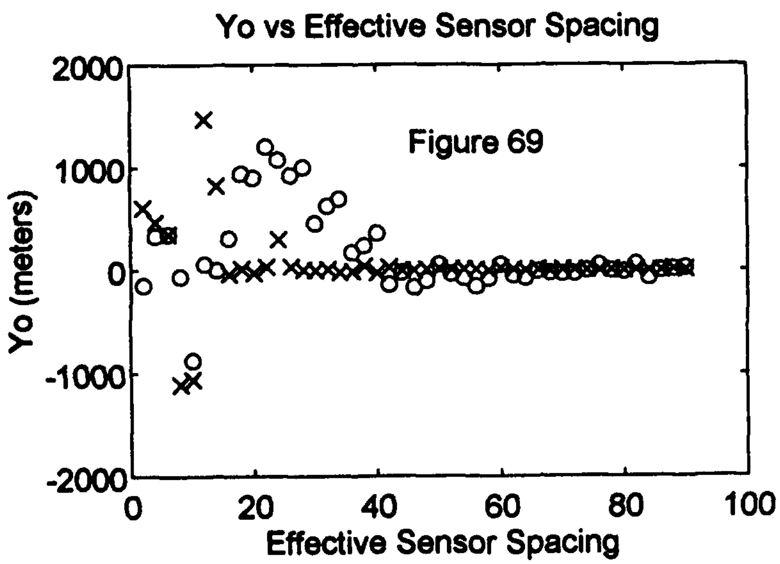

Standard Deviation of YT
x 10^4
Figure 64
meters
4
3
2
1
0
0
10
20
30
40
50
n

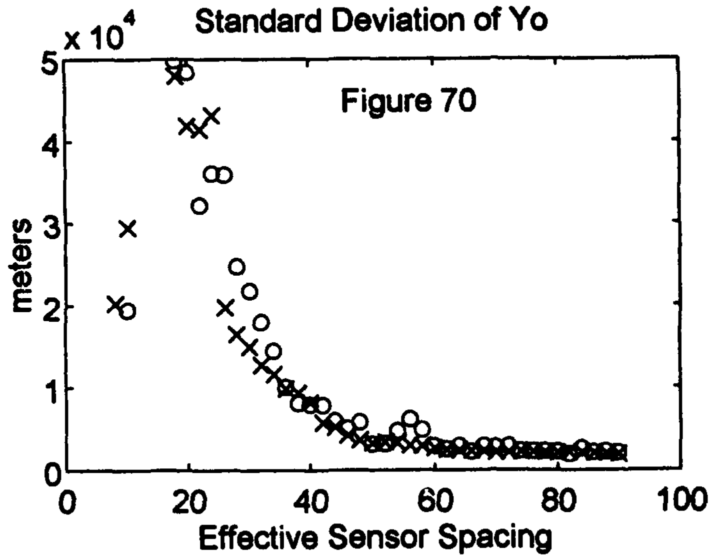

Projected Current Position ZT
Figure 65
ZT (meters)
0
-1000
-2000
-3000
-4000
-5000
0
10
20
30
40
50
n

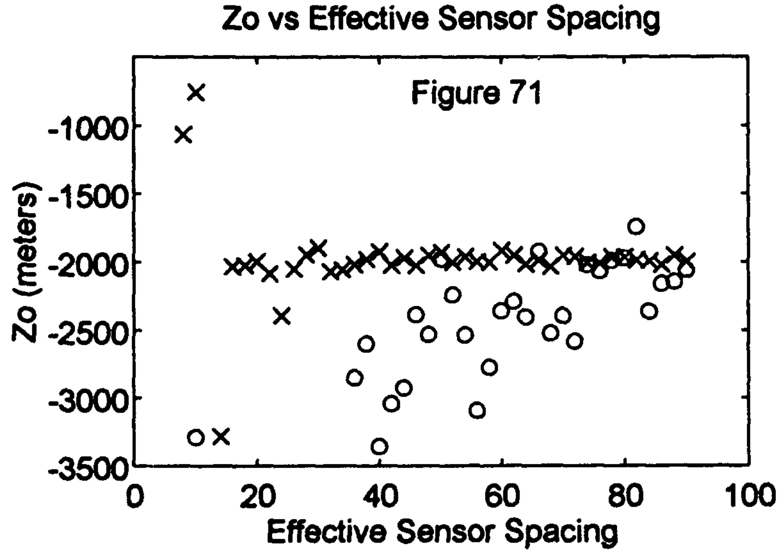

Standard Deviation of ZT
x 10^4
Figure 66
meters
8
6
4
2
0
0
10
20
30
40
50
n


Xo vs Effective Sensor Spacing
Figure 67
Xo (meters)
1000
500
0
-500
-1000
0
20
40
60
80
100
Effective Sensor Spacing

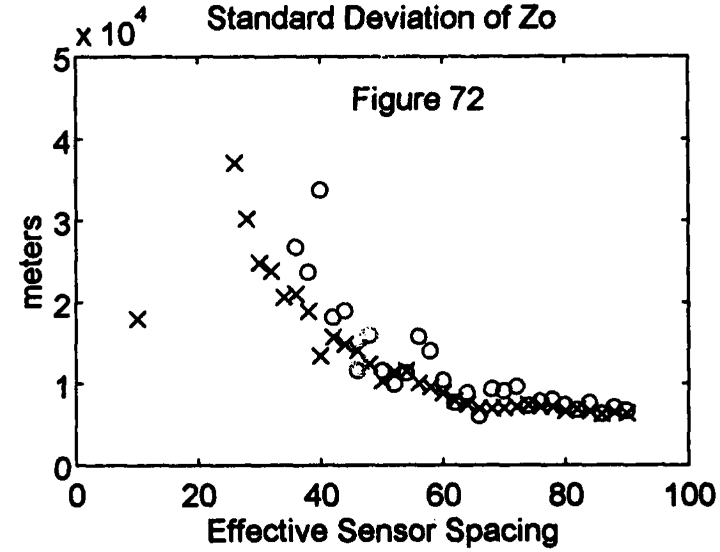

Standard Deviation of Xo
x 10^4
Figure 68
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing


Yo vs Effective Sensor Spacing
Figure 69
Yo (meters)
2000
1000
0
-1000
-2000
0
20
40
60
80
100
Effective Sensor Spacing

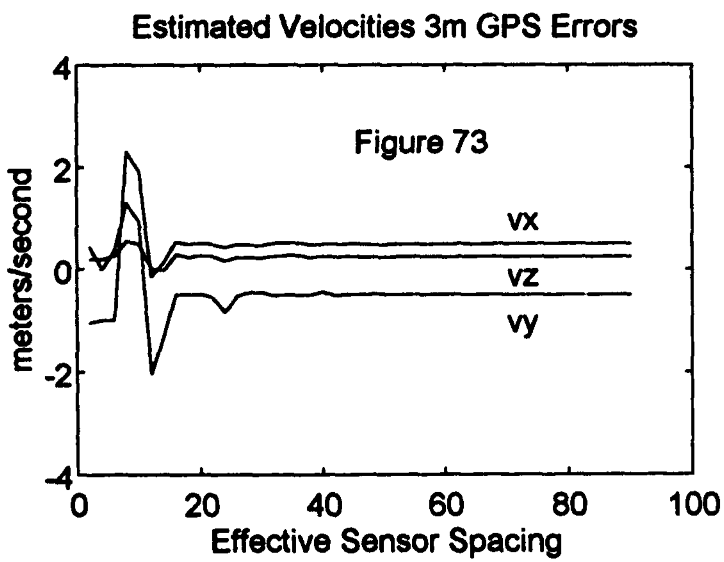

Standard Deviation of Yo
x 10^4
Figure 70
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing



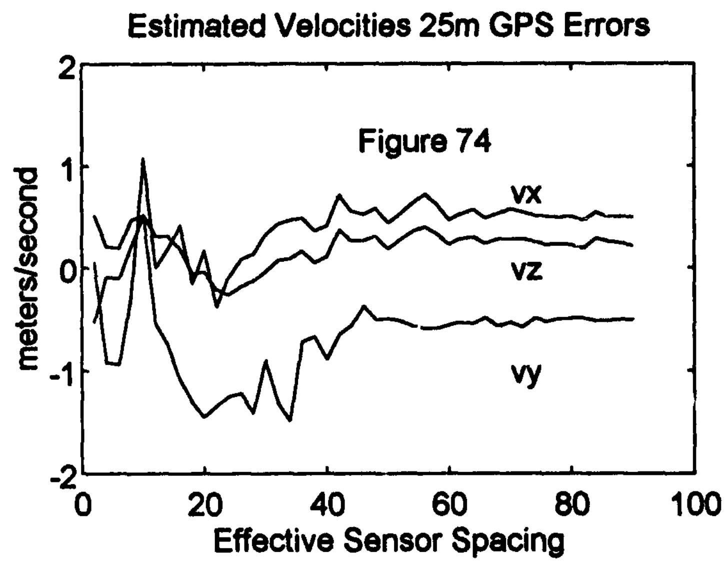
Zo vs Effective Sensor Spacing
Figure 71
Zo (meters)
-1000
-1500
-2000
-2500
-3000
-3500
0
20
40
60
80
100
Effective Sensor Spacing



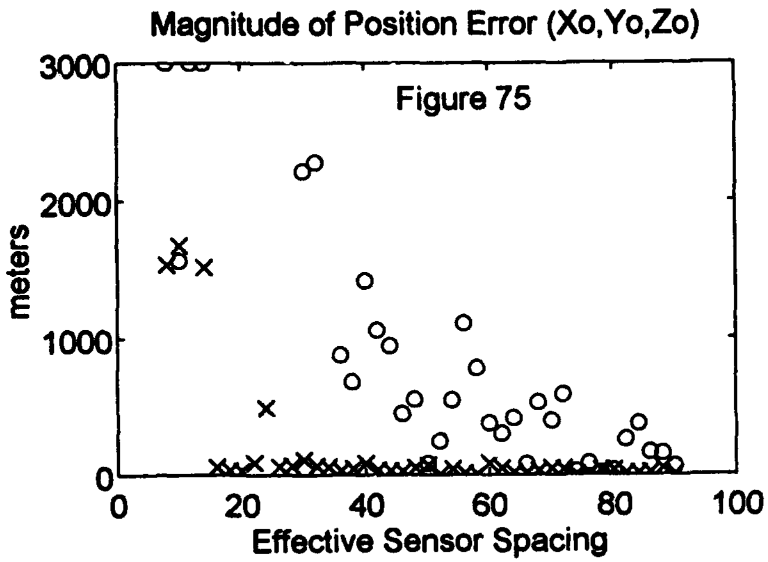
Standard Deviation of Zo
x 10^4
Figure 72
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

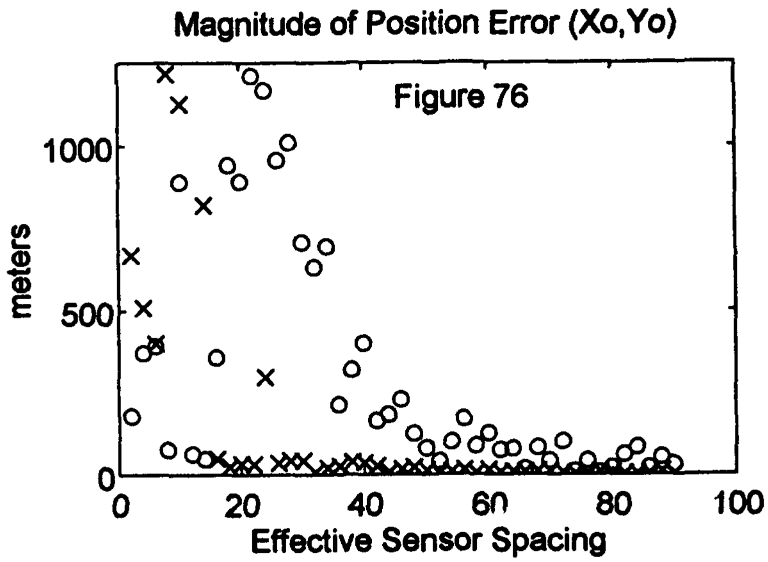


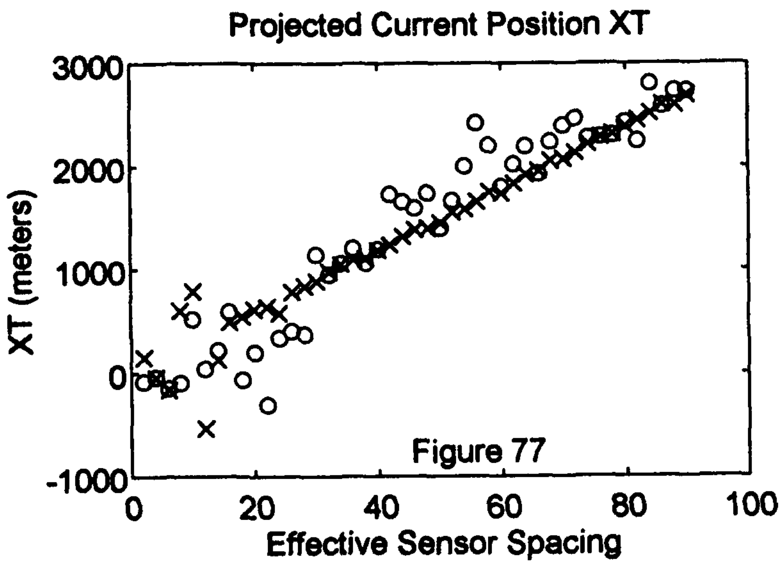
Estimated Velocities 3m GPS Errors
Figure 73
meters/second
vx
vz
vy
4
2
0
-2
-4
0
20
40
60
80
100
Effective Sensor Spacing

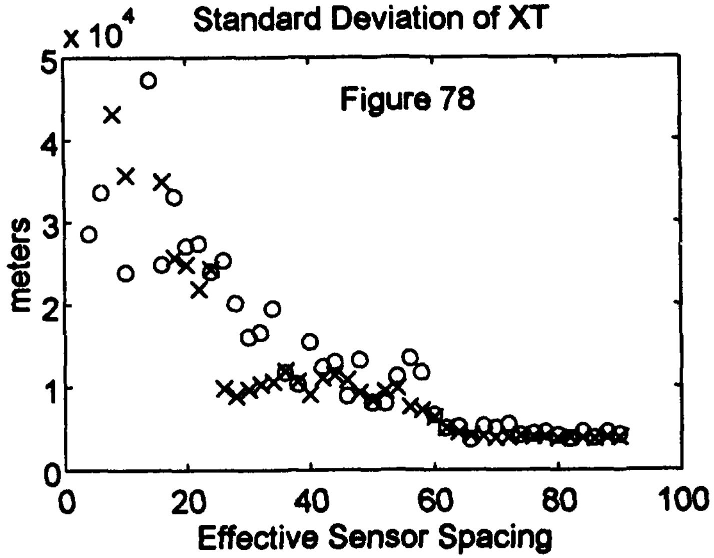


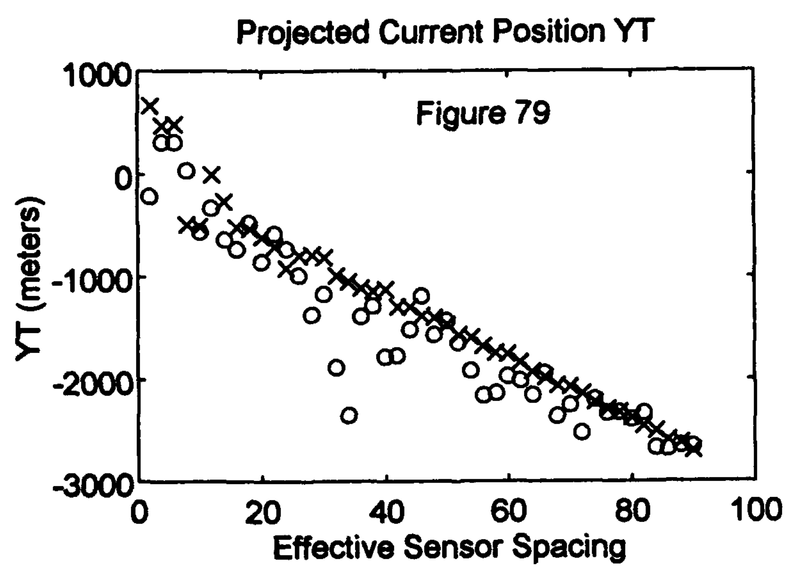
Estimated Velocities 25m GPS Errors
Figure 74
meters/second
vx
vz
vy
2
1
0
-1
-2
0
20
40
60
80
100
Effective Sensor Spacing

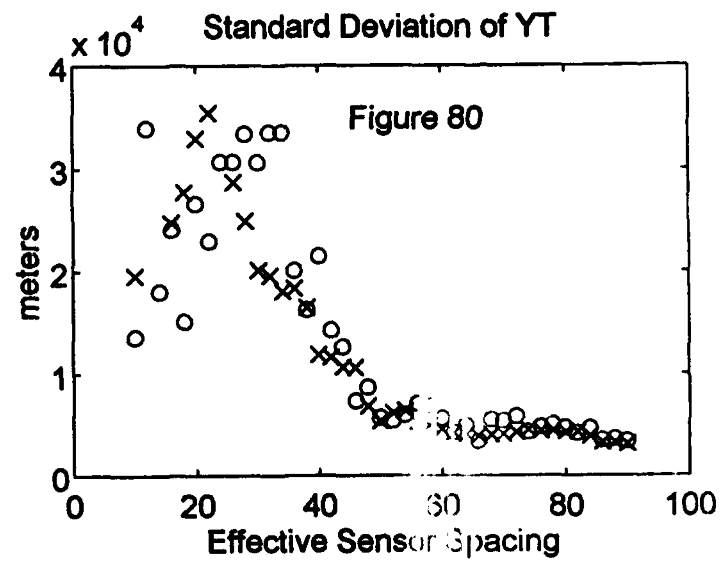

Magnitude of Position Error (Xo,Yo,Zo)
Figure 75
meters
3000
2000
1000
0
0
20
40
60
80
100
Effective Sensor Spacing

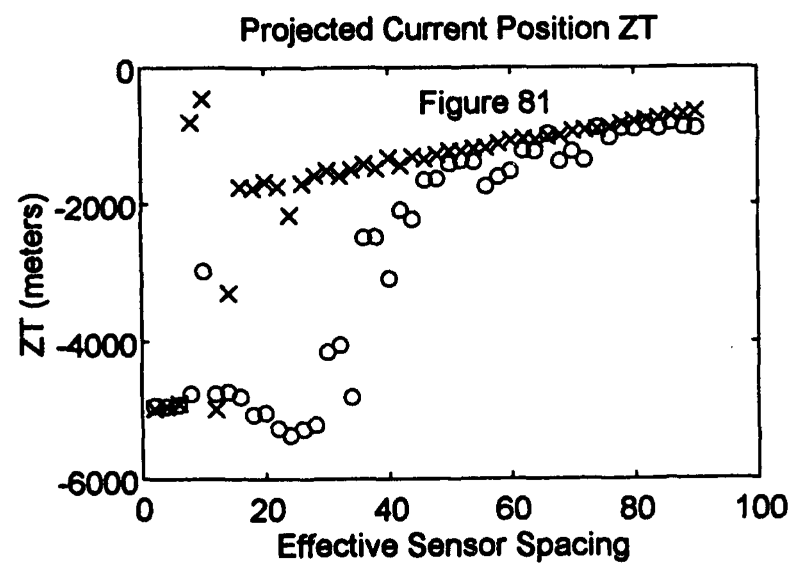

Magnitude of Position Error (Xo,Yo)
Figure 76
meters
1000
500
0
0
20
40
60
80
100
Effective Sensor Spacing

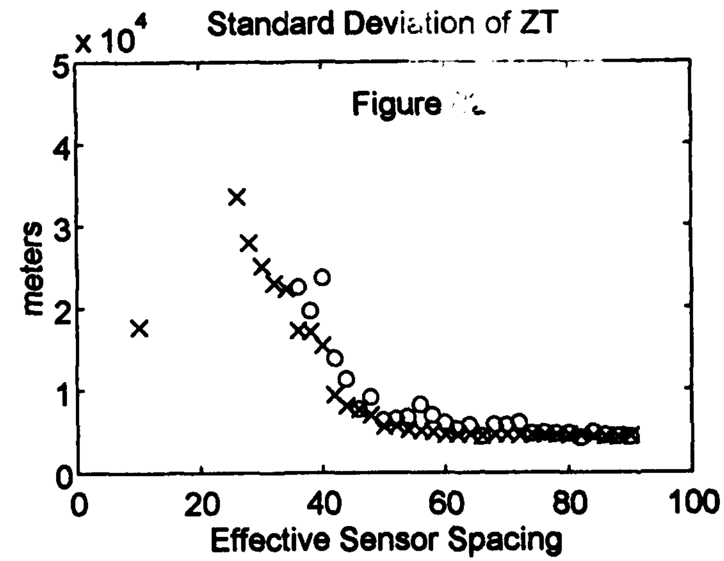

Projected Current Position XT
Figure 77
XT (meters)
3000
2000
1000
0
-1000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of XT
x 10^4
Figure 78
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing


Projected Current Position YT
Figure 79
YT (meters)
1000
0
-1000
-2000
-3000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of YT
x 10^4
Figure 80
meters
4
3
2
1
0
0
20
40
80
100
Effective Sensor Spacing


Projected Current Position ZT
Figure 81
ZT (meters)
0
-2000
-4000
-6000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of ZT
x 10^4
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

4.4.2 Scenario 5 - Mov[illegible] Acoustic [illegible] Located at:

[illegible]

Fi[illegible] [illegible]ow the results of the maximum likelihood estimation algo[illegible] [illegible] different numbers of experiment vectors, and figures 4-99 through 4-114 show the results of the maximum likelihood estimation algorithm for different effective sensor spacings for scenario 5. Similar to the previous scenarios, we see that as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, while the standard deviation of the estimated position and velocities decrease. We also see that as the effective sensor spacing increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, and the standard deviation of the estimated position decreases. As before, the position and velocity errors initially decrease rapidly, however after the number of experiment vectors used reaches approximately 30, and the effective sensor spacing reaches approximately 45, the errors and the standard deviations decrease much slower.

Figures 4-93 through 4-98 and figures 4-109 through 4-114 show the estimation algorithm's ability to track the linearly moving acoustic beacon for this scenario. We see that the projected current positions follow the linearly moving acoustic beacon fairly well for both 3 meter and 25 meter GPS errors. This is because the estimated initial positions x_o, y_o, and z_o obtained using 25 meter GPS errors were better in this scenario then for scenario 4. In general this is not the case, however the measurement errors in this scenario simply led to a better estimate of the acoustic beacon's position.

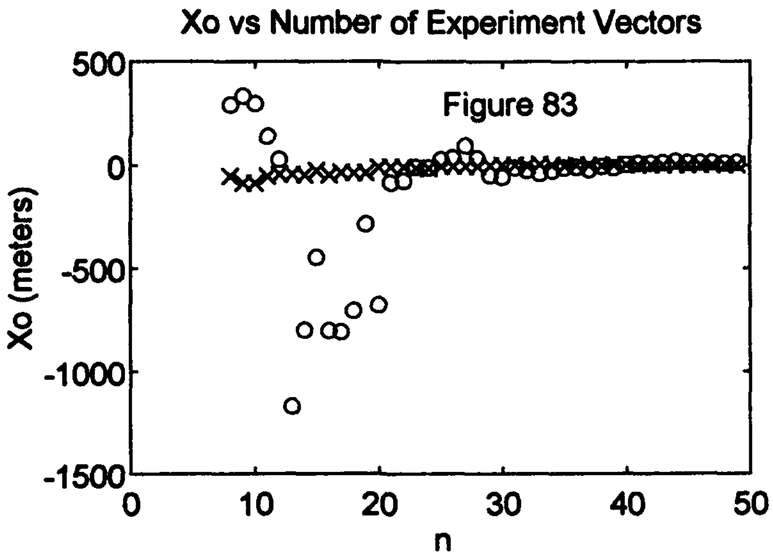


Figure 83

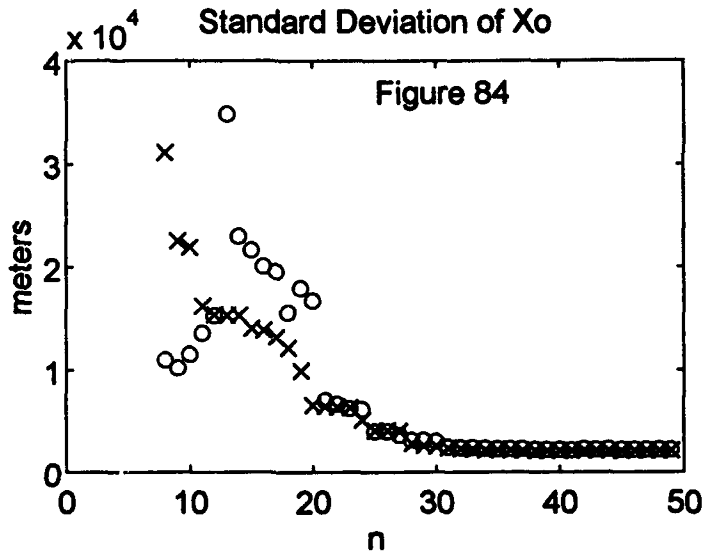


Figure 84

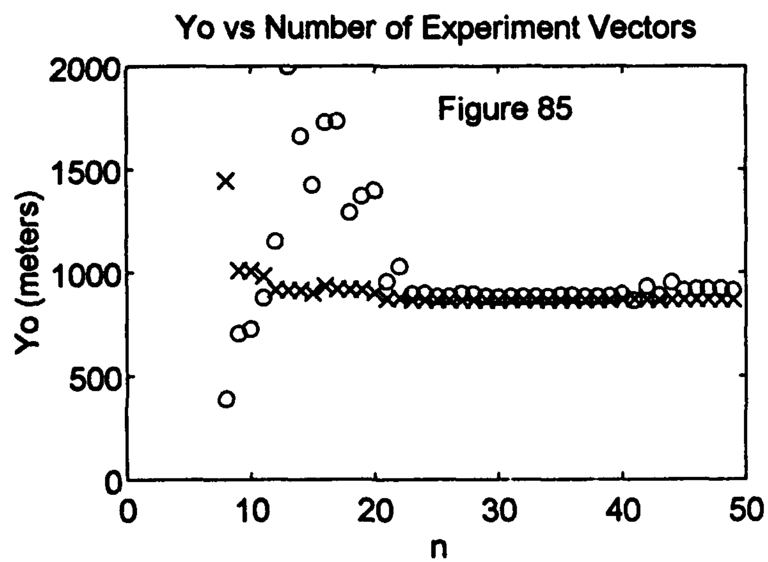


Figure 85

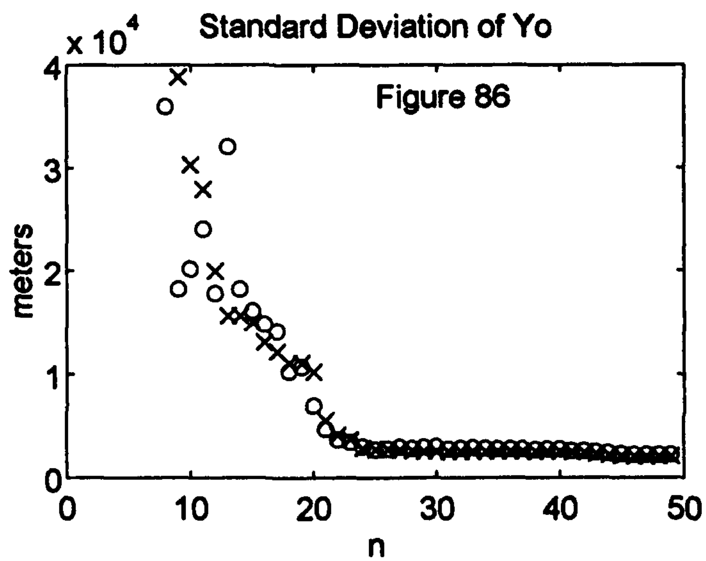
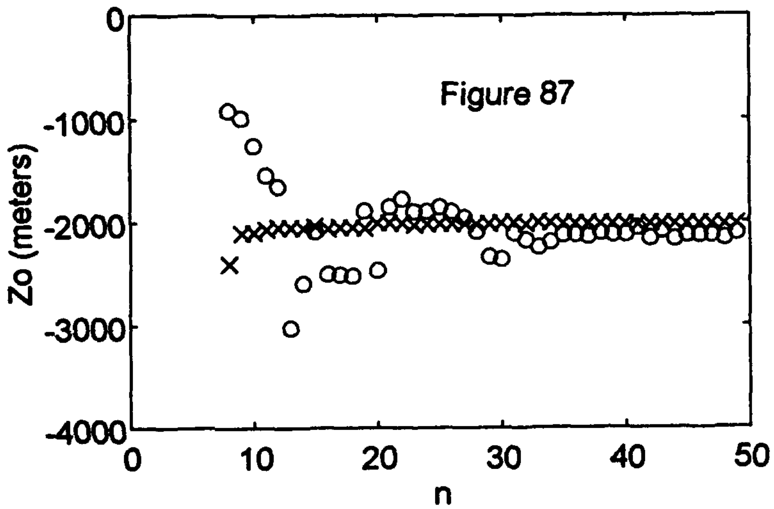
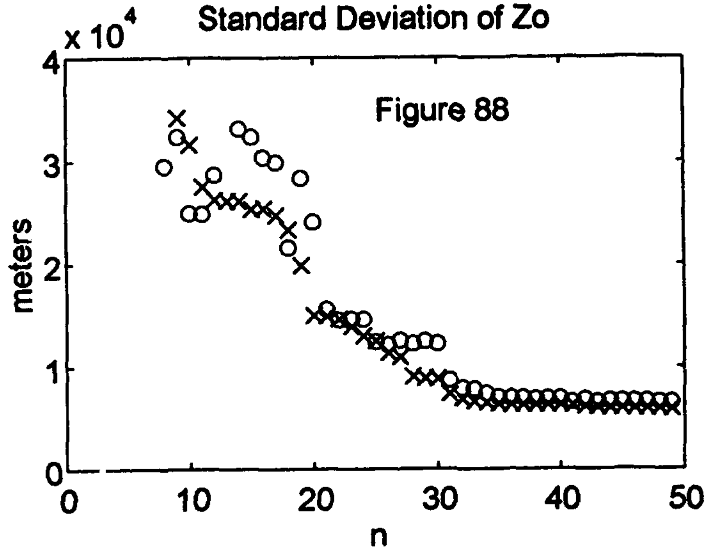


Figure 86

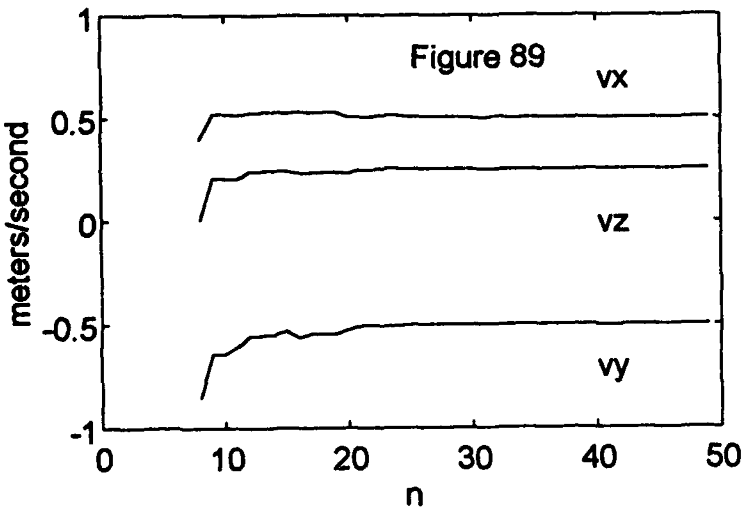
Zo vs Number of Experiment Vectors



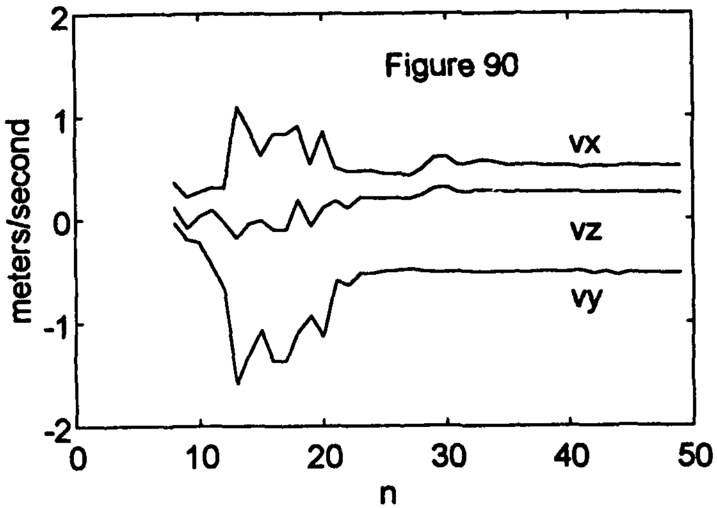
Standard Deviation of Zo

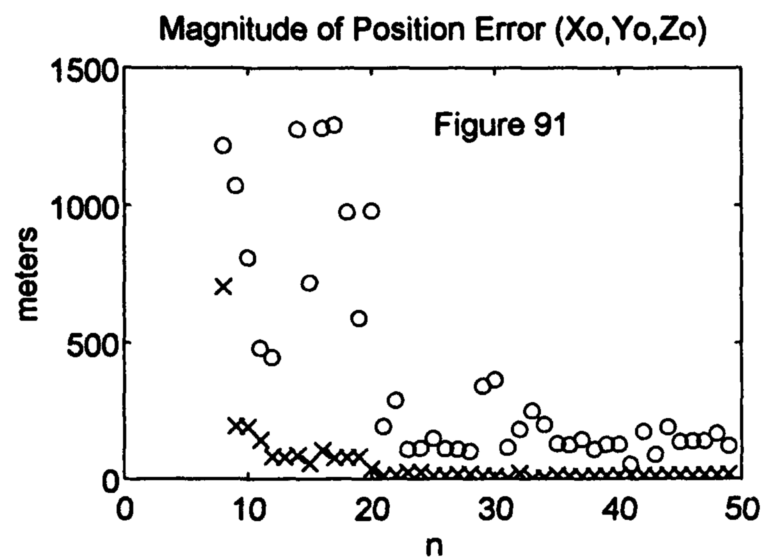


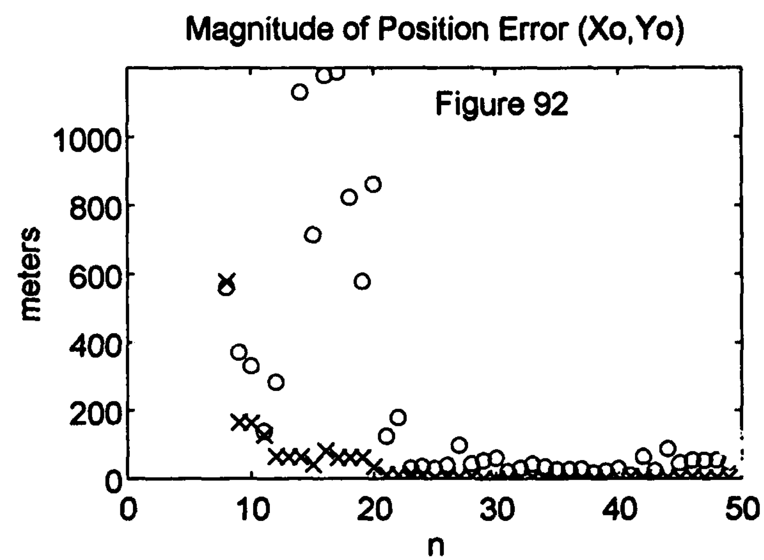
Estimated Velocities 3m GPS Errors

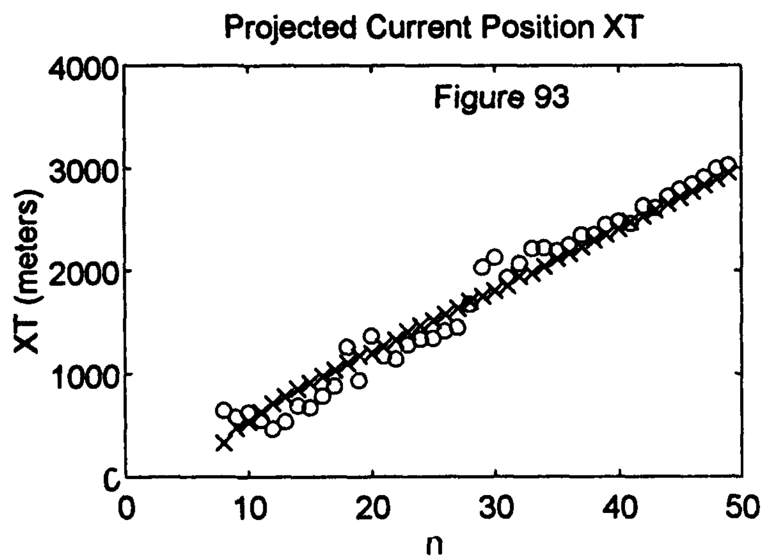


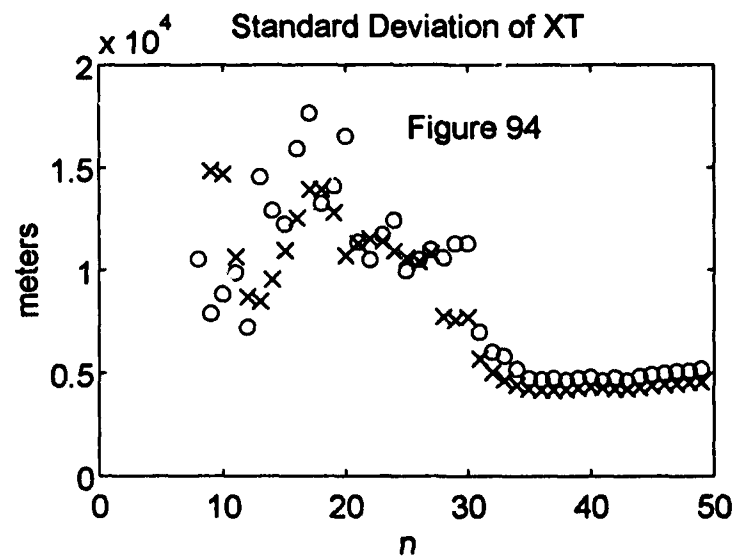
Estimated Velocities 25m GPS Errors

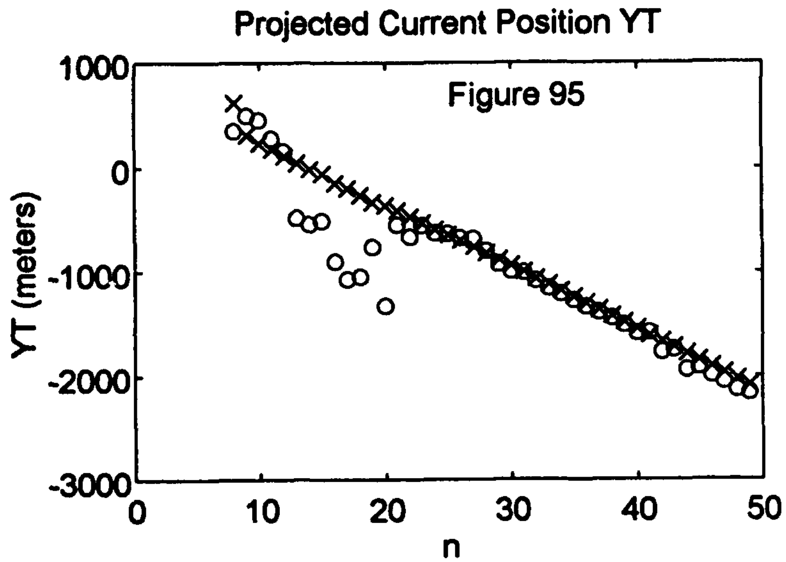


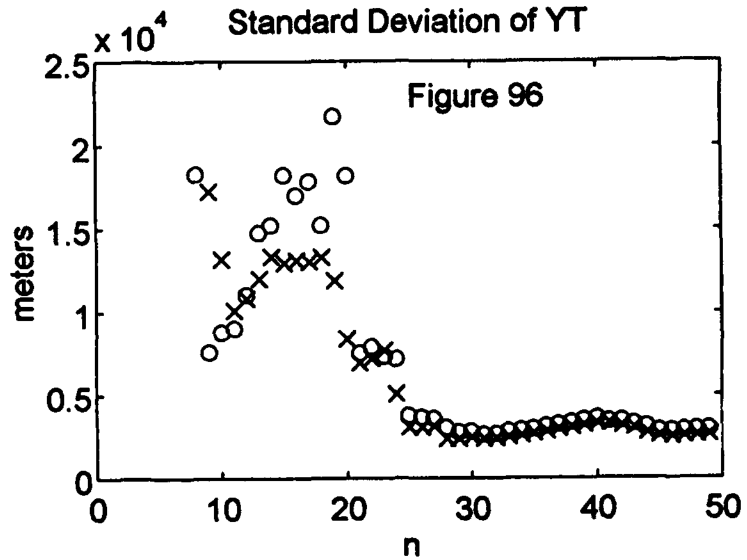

Magnitude of Position Error (Xo,Yo,Zo)
Figure 91
meters
1500
1000
500
0
0
10
20
30
40
50
n

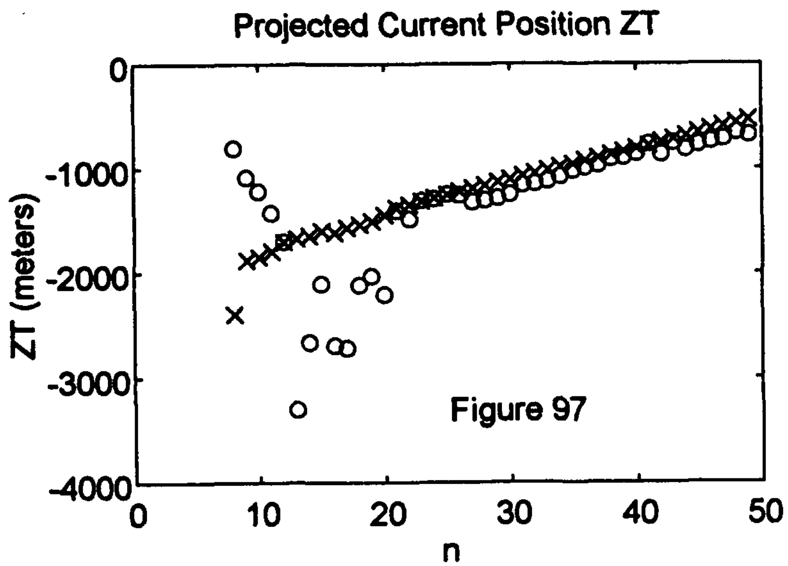

Magnitude of Position Error (Xo,Yo)
Figure 92
meters
1000
800
600
400
200
0
0
10
20
30
40
50
n

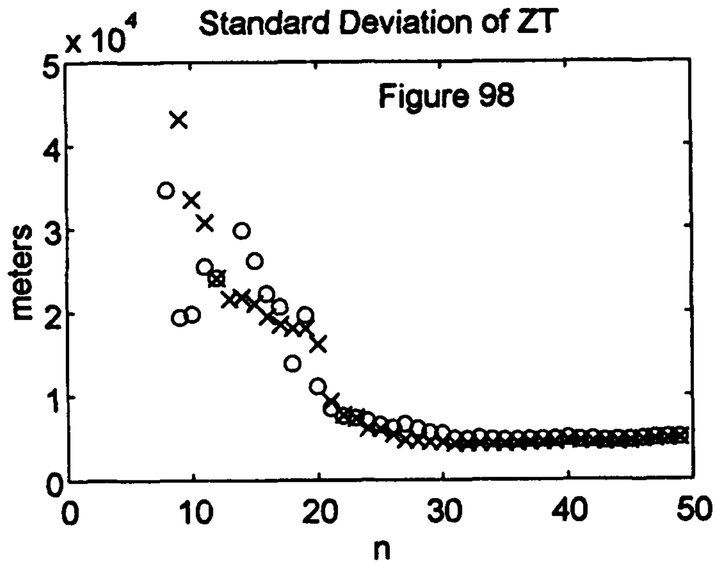

Projected Current Position XT
Figure 93
XT (meters)
4000
3000
2000
1000
C
0
10
20
30
40
50
n

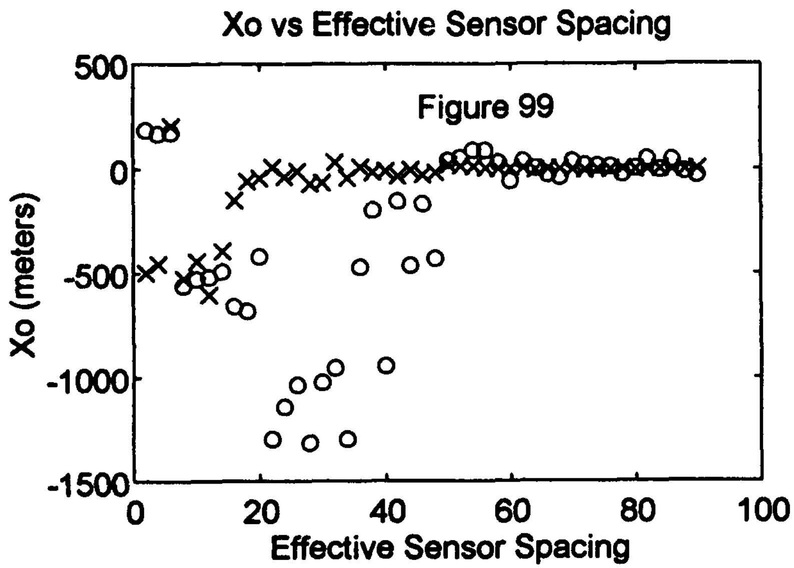

Standard Deviation of XT
Figure 94
x 10^4
meters
2
1.5
1
0.5
0
0
10
20
30
40
50
n

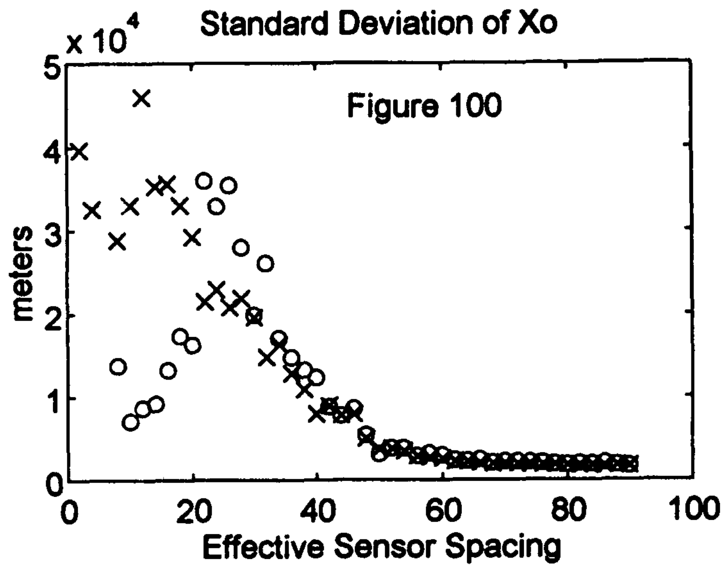

Projected Current Position YT
Figure 95
YT (meters)
1000
0
-1000
-2000
-3000
0
10
20
30
40
50
n

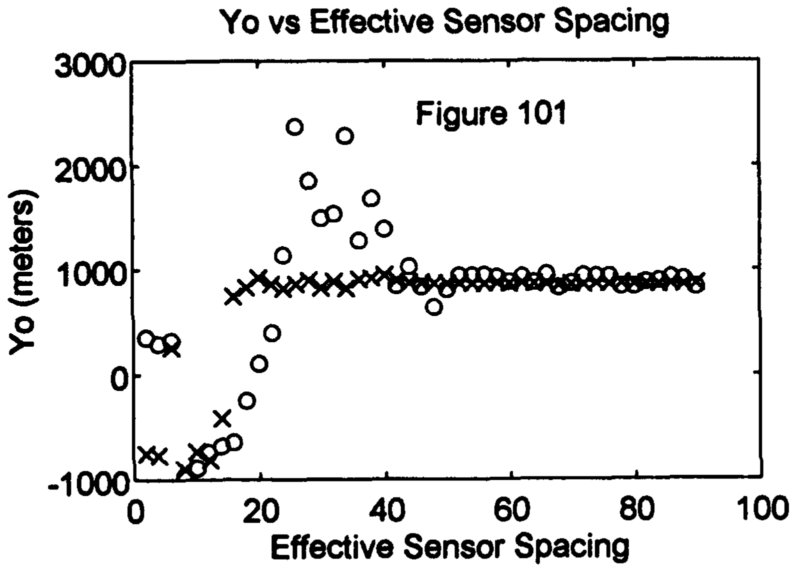

Standard Deviation of YT
x 10^4
Figure 96
meters
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
n

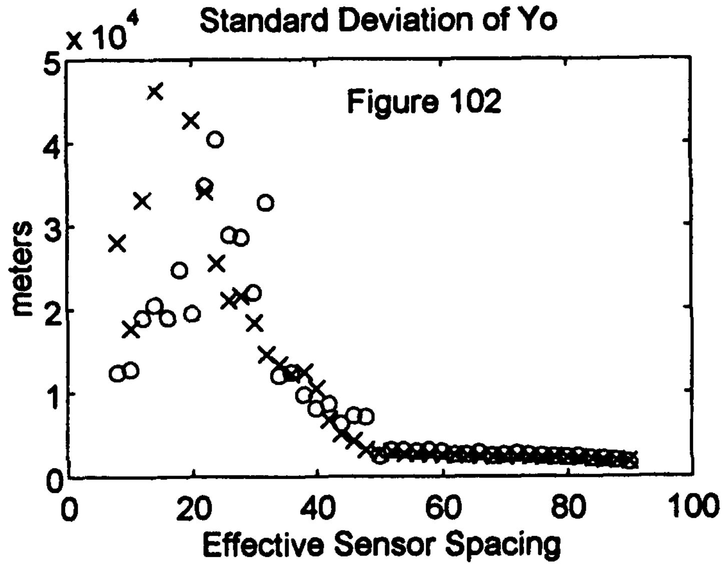

Projected Current Position ZT
Figure 97
ZT (meters)
0
-1000
-2000
-3000
-4000
0
10
20
30
40
50
n

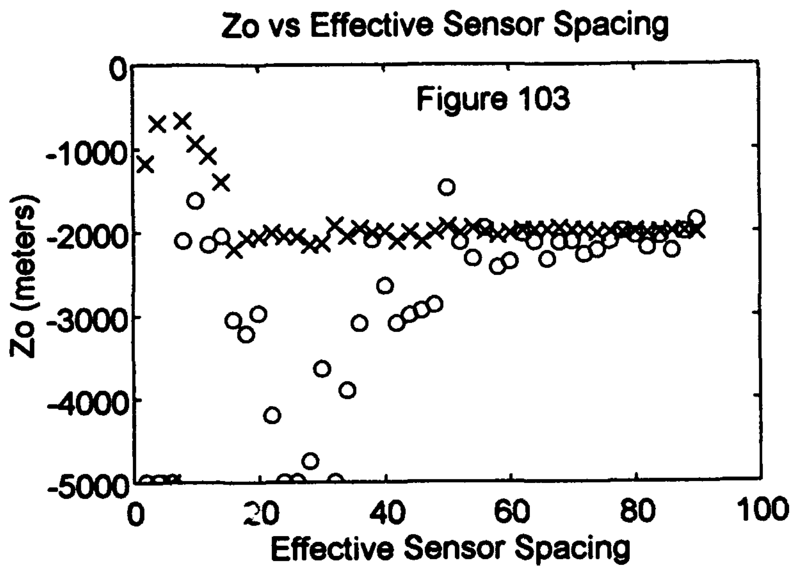

Standard Deviation of ZT
x 10^4
Figure 98
meters
5
4
3
2
1
0
0
10
20
30
40
50
n

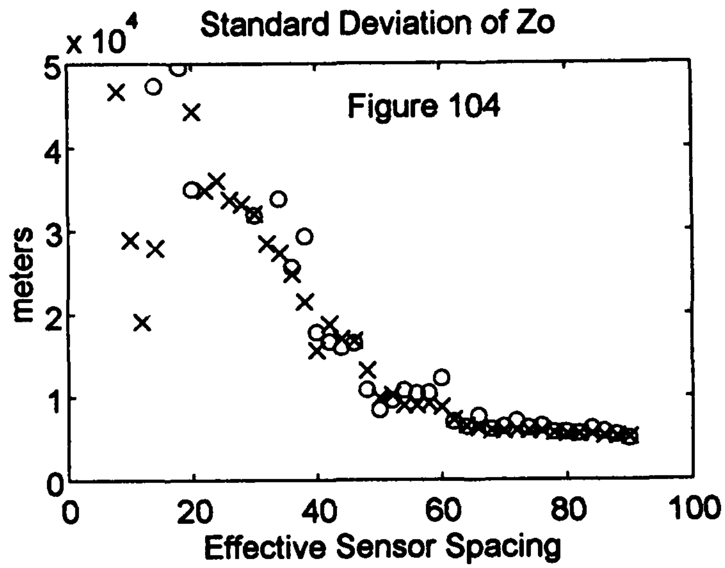

Xo vs Effective Sensor Spacing
Figure 99
Xo (meters)
500
0
-500
-1000
-1500
0
20
40
60
80
100
Effective Sensor Spacing

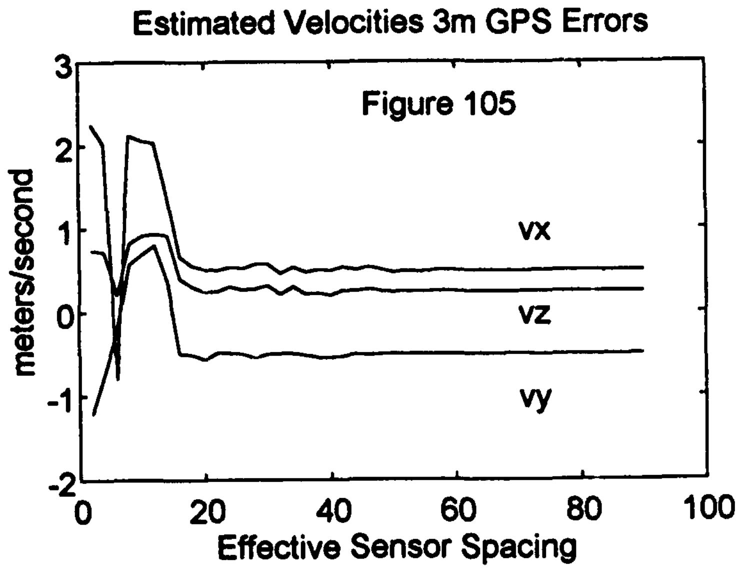

Standard Deviation of Xo
x 10^4
Figure 100
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

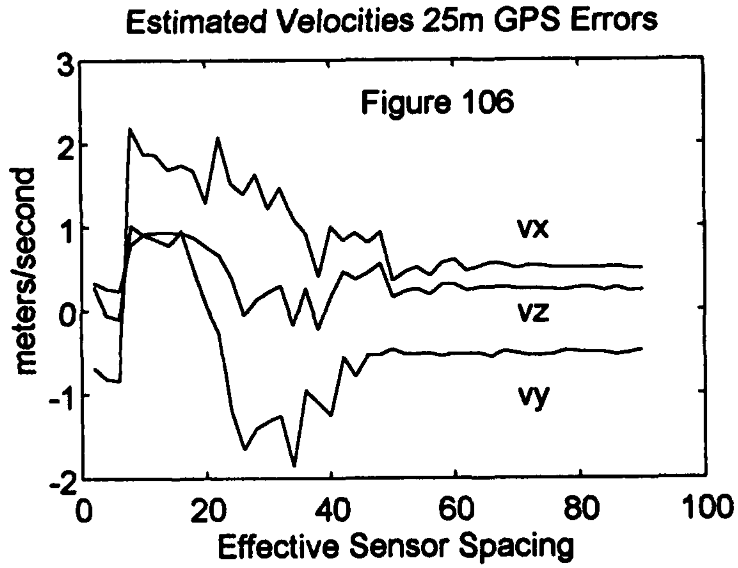

Yo vs Effective Sensor Spacing
Figure 101
Yo (meters)
3000
2000
1000
0
-1000
0
20
40
60
80
100
Effective Sensor Spacing

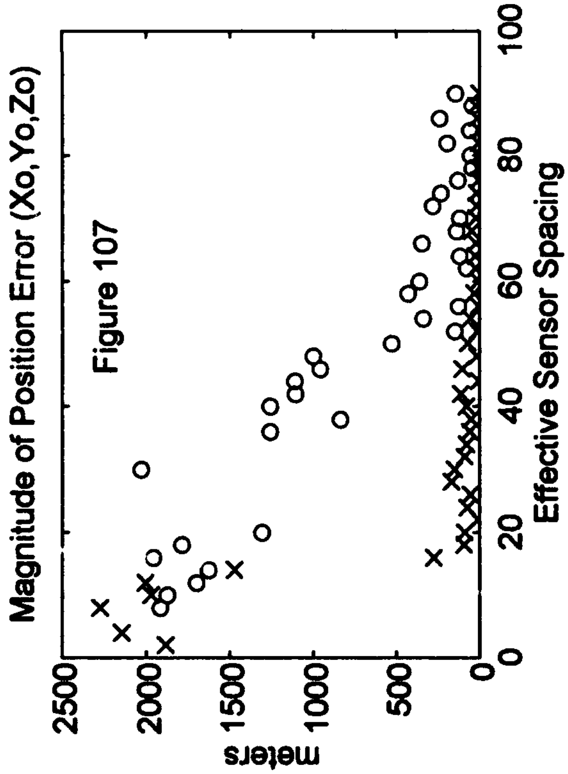

Standard Deviation of Yo
x 10^4
Figure 102
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

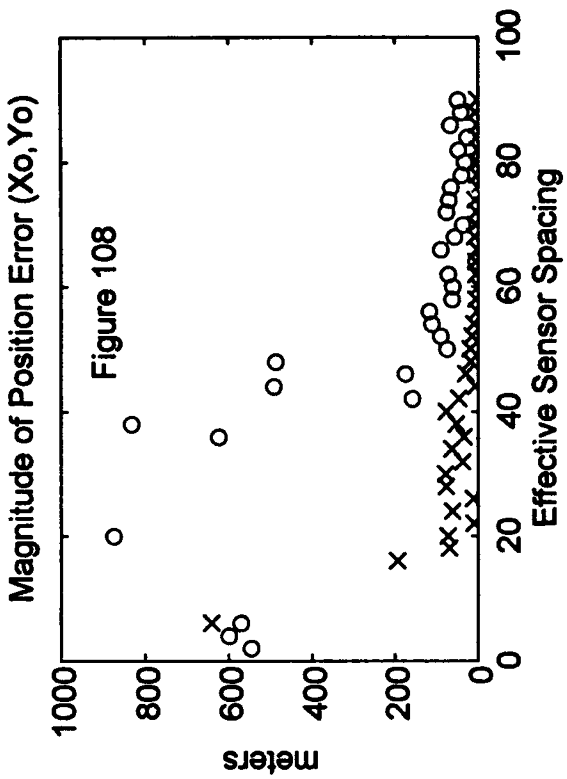

Zo vs Effective Sensor Spacing
Figure 103
Zo (meters)
0
-1000
-2000
-3000
-4000
-5000
0
20
40
60
80
100
Effective Sensor Spacing

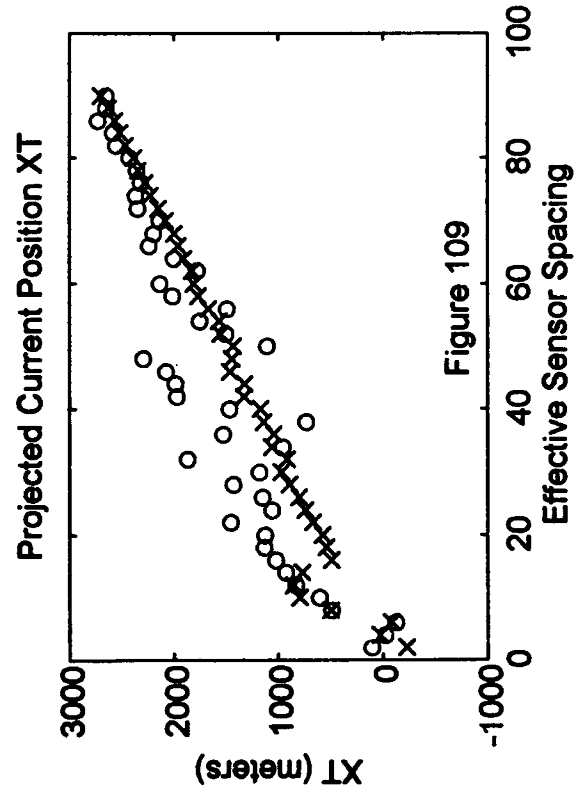

Standard Deviation of Zo
x 10^4
Figure 104
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

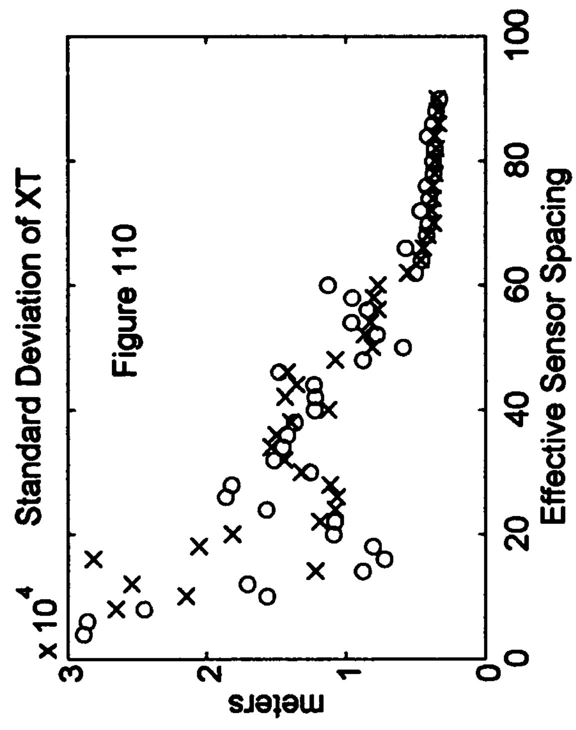

Estimated Velocities 3m GPS Errors
Figure 105
meters/second
3
2
1
0
-1
-2
vx
vz
vy
0
20
40
60
80
100
Effective Sensor Spacing


Estimated Velocities 25m GPS Errors
Figure 106
meters/second
3
2
1
0
-1
-2
vx
vz
vy
0
20
40
60
80
100
Effective Sensor Spacing

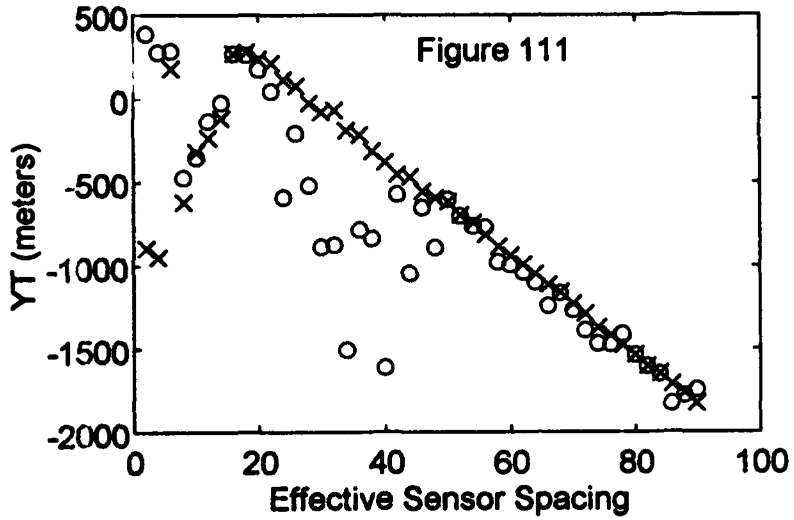

Magnitude of Position Error (Xo,Yo,Zo)
Figure 107
meters
2500
2000
1500
1000
500
0
0
20
40
60
80
100
Effective Sensor Spacing


Magnitude of Position Error (Xo,Yo)
Figure 108
meters
1000
800
600
400
200
0
0
20
40
60
80
100
Effective Sensor Spacing

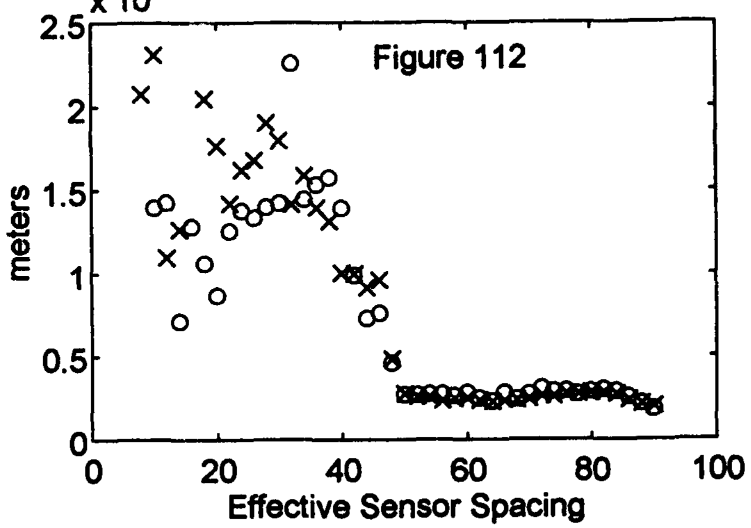

Projected Current Position XT
Figure 109
XT (meters)
3000
2000
1000
0
-1000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of XT
Figure 110
x 10
4
meters
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

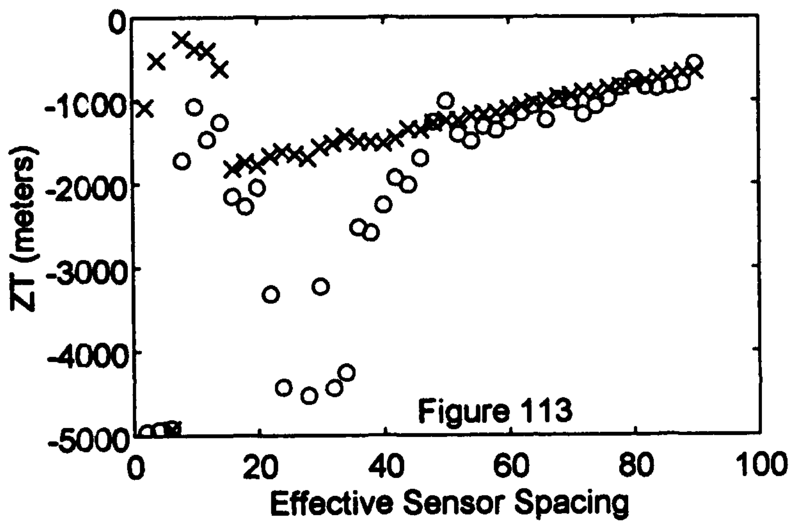
Projected Current Position YT



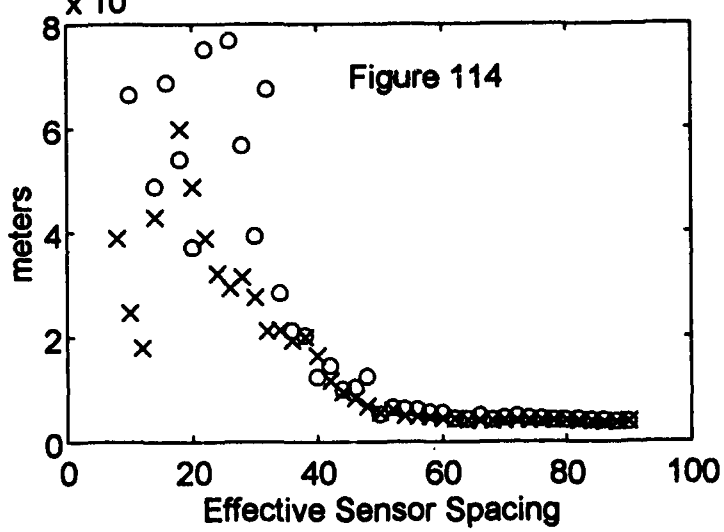
Standard Deviation of YT



Projected Current Position ZT



Standard Deviation of ZT

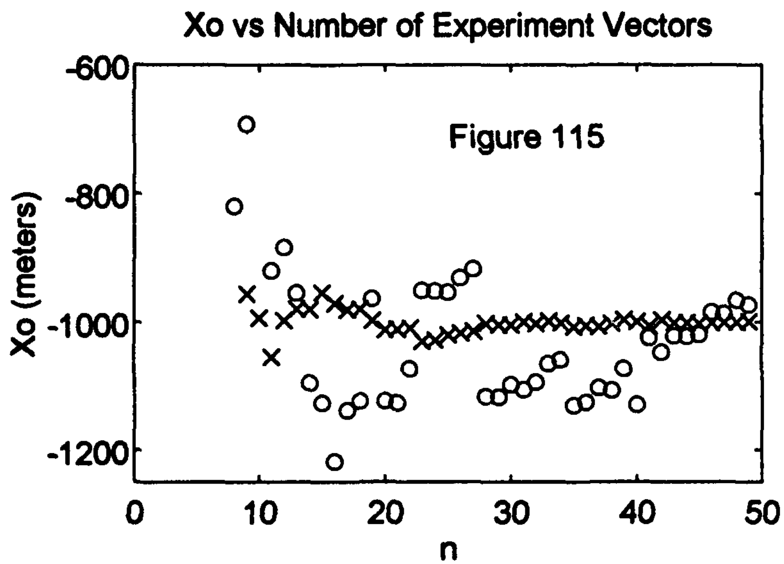


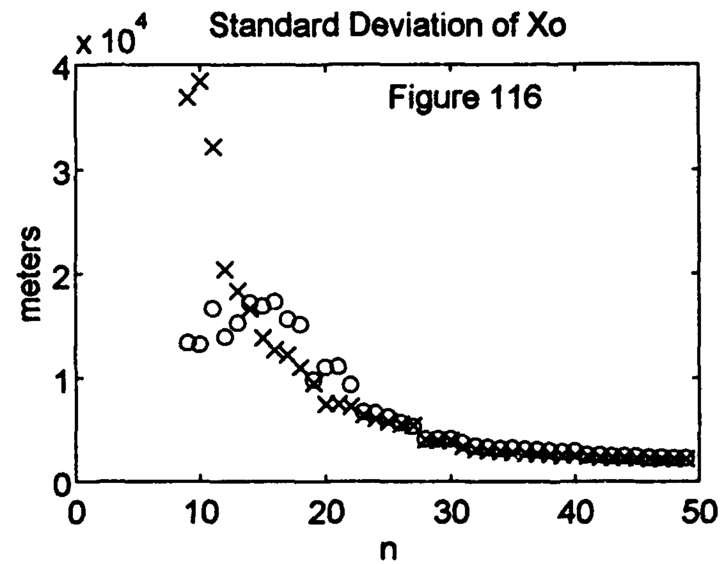
4.4.3 Scenario 6 - Moving Acoustic Beacon Located at:

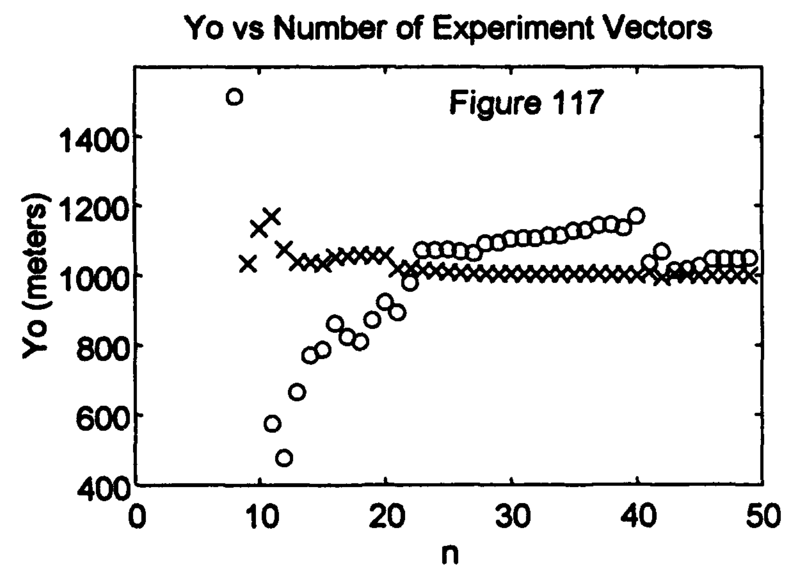
$\theta = [-10000\ 1000\ -2000\ 0.5\ -0.5\ 0.25]^T$

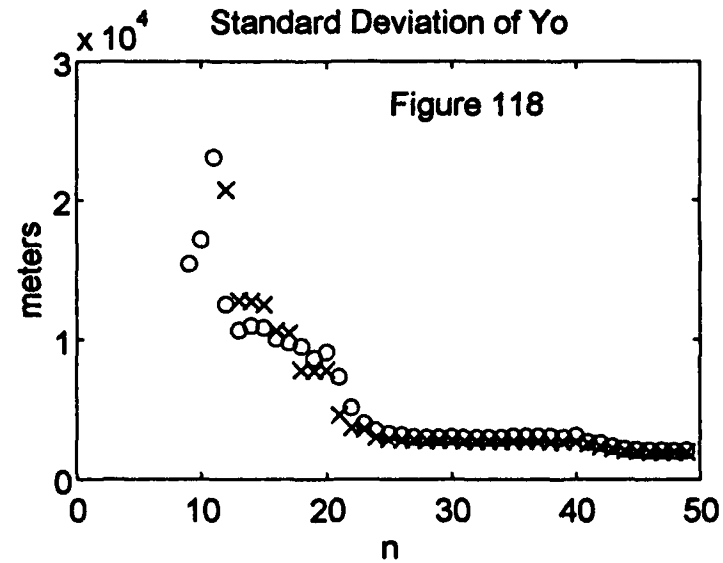
Figures 4-115 through 4-130 show the results of the maximum likelihood estimation algorithm for different numbers of experiment vectors, and figures 4-131 through 4-146 show the results of the maximum likelihood estimation algorithm for different effective sensor spacings for scenario 6. Again we see that as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, and the standard deviation of the estimated position and velocities decrease. We also see that as the effective sensor spacing increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, while the standard deviation of the estimated position decreases. As before, the position and velocity errors initially decrease rapidly, however after the number of experiment vectors used reaches approximately 30, or the effective sensor spacing reaches approximately 45, the errors and the standard deviations decrease much slower.

Figures 4-125 through 4-130 and figures 4-141 through 4-146 show the estimation algorithm's ability to track the linearly moving acoustic beacon for this scenario. We see that the projected current positions follow the linearly moving acoustic beacon fairly well using 3 meter GPS errors. Again, because of the errors in estimating the acoustic beacon's initial positions x_o, y_o, and z_o, the projected current position obtained from using 25 meter GPS errors does not track the acoustic beacon's motion as well.

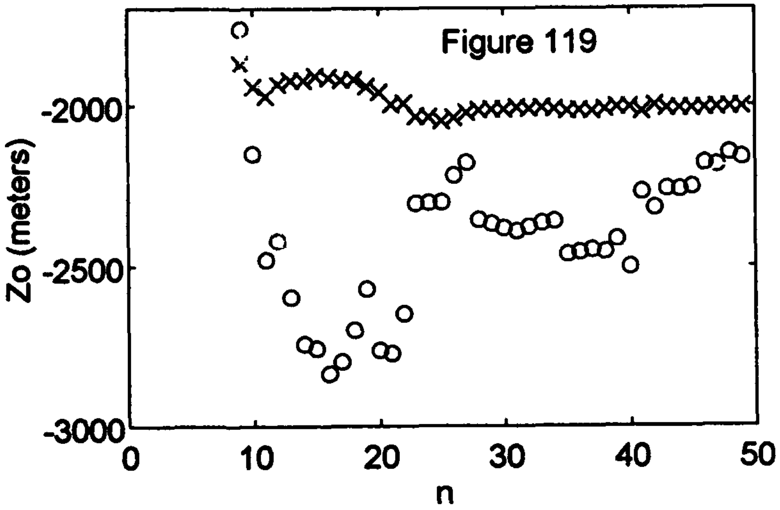

Xo vs Number of Experiment Vectors
Figure 115
Xo (meters)
-600
-800
-1000
-1200
0
10
20
30
40
50
n


Standard Deviation of Xo
x 10^4
Figure 116
meters
4
3
2
1
0
0
10
20
30
40
50
n

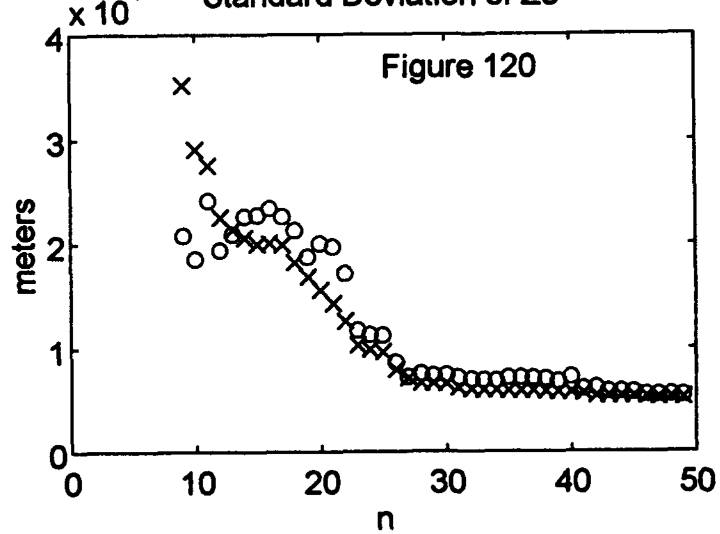

Yo vs Number of Experiment Vectors
Figure 117
Yo (meters)
1400
1200
1000
800
600
400
0
10
20
30
40
50
n


Standard Deviation of Yo
x 10^4
Figure 118
meters
3
2
1
0
0
10
20
30
40
50
n

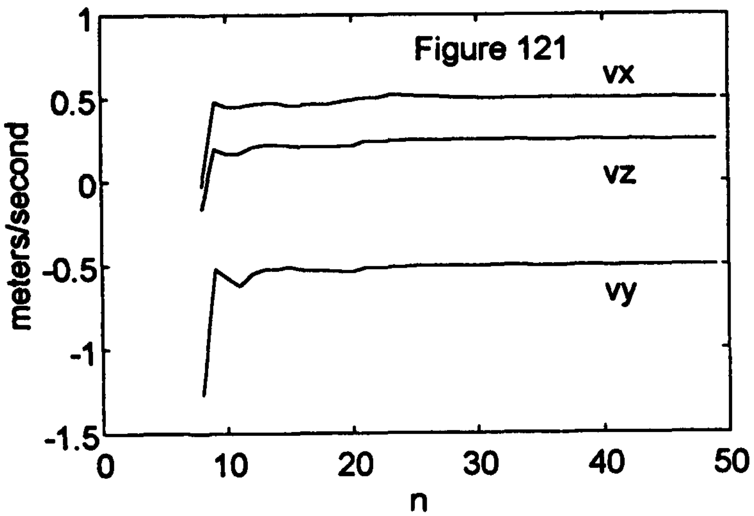
Zo vs Number of Experiment Vectors



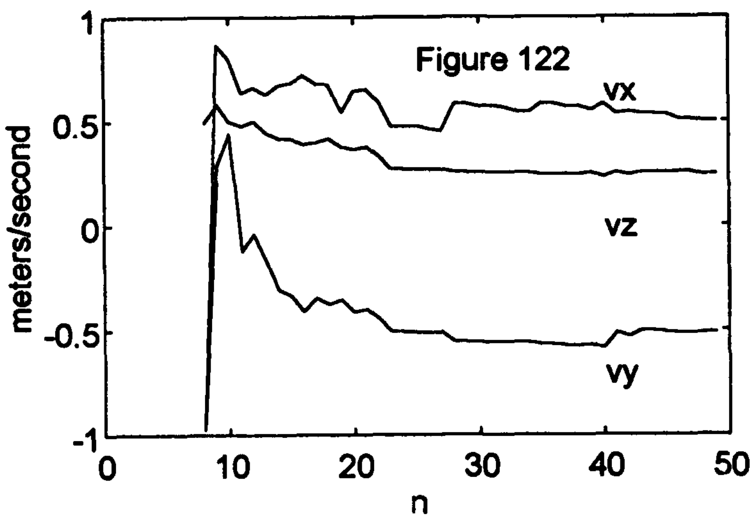
Standard Deviation of Zo

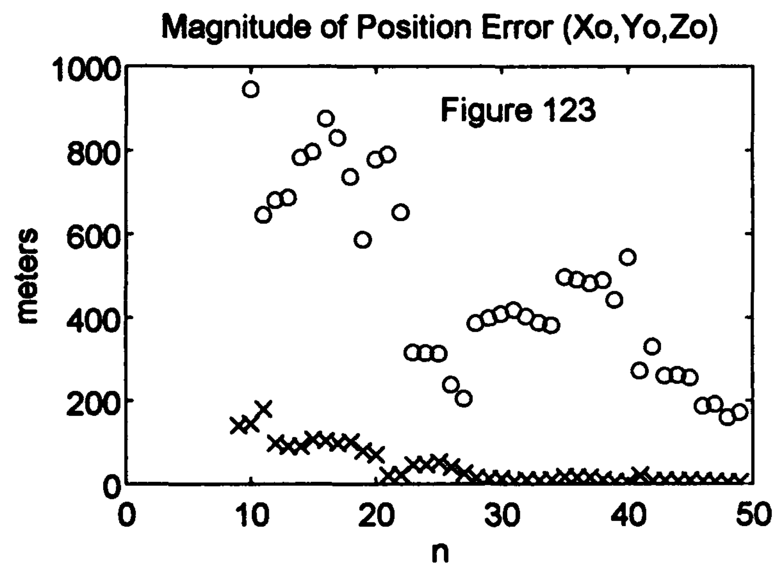


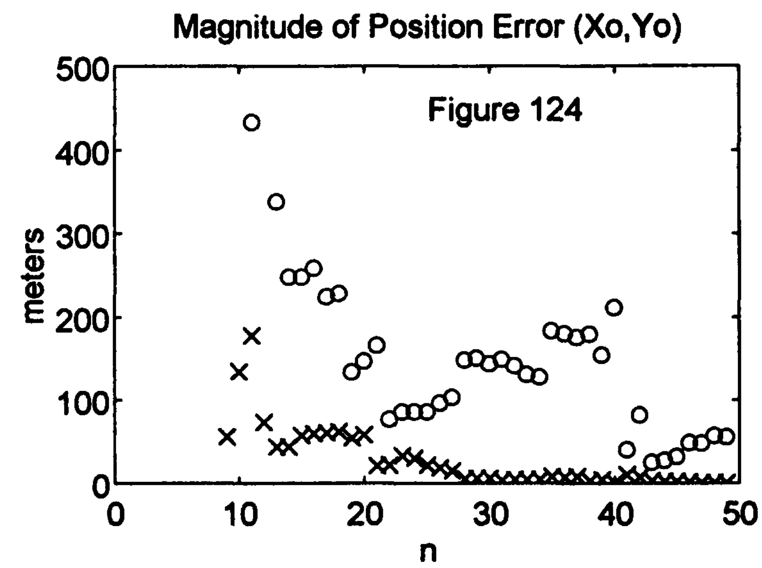
Estimated Velocities 3m GPS Errors

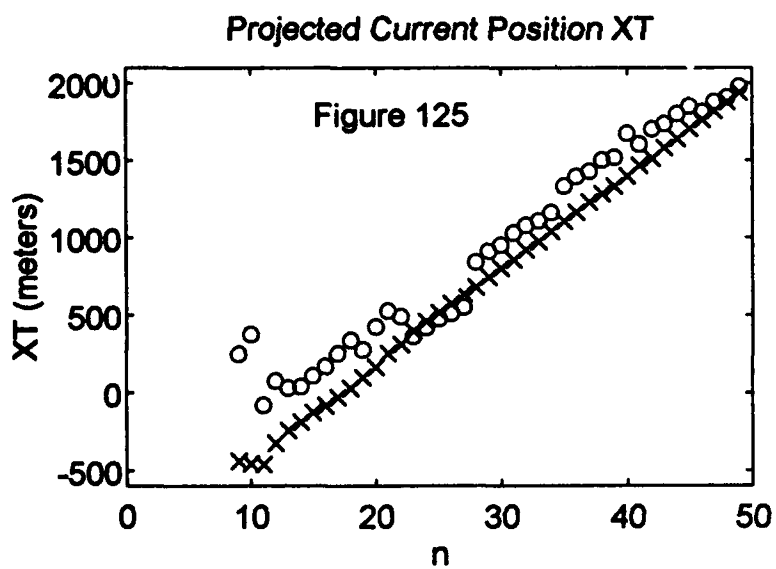


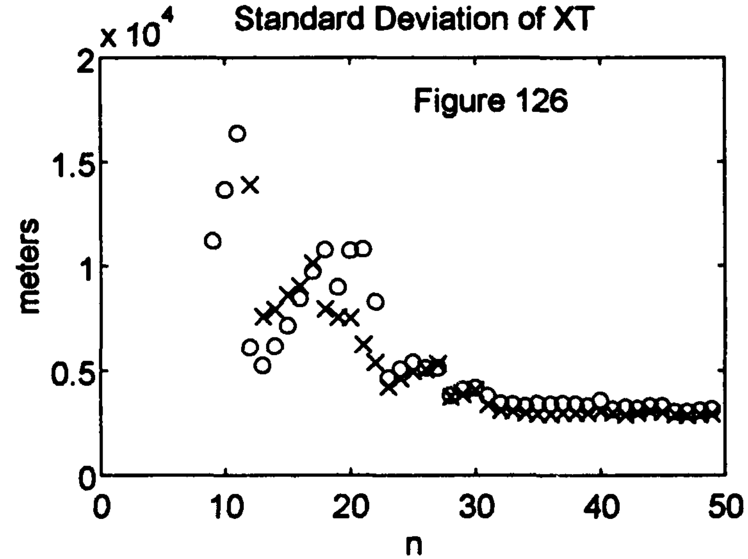
Estimated Velocities 25m GPS Errors

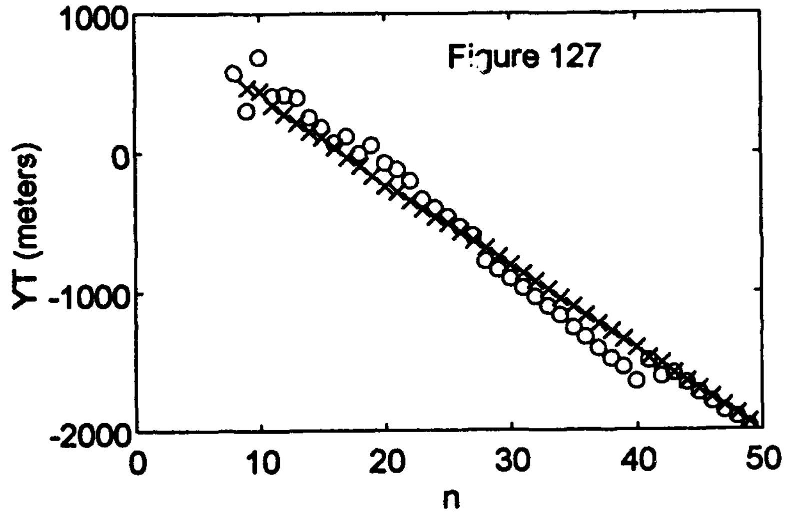



Magnitude of Position Error (Xo,Yo,Zo)
Figure 123
meters
1000
800
600
400
200
0
0
10
20
30
40
50
n

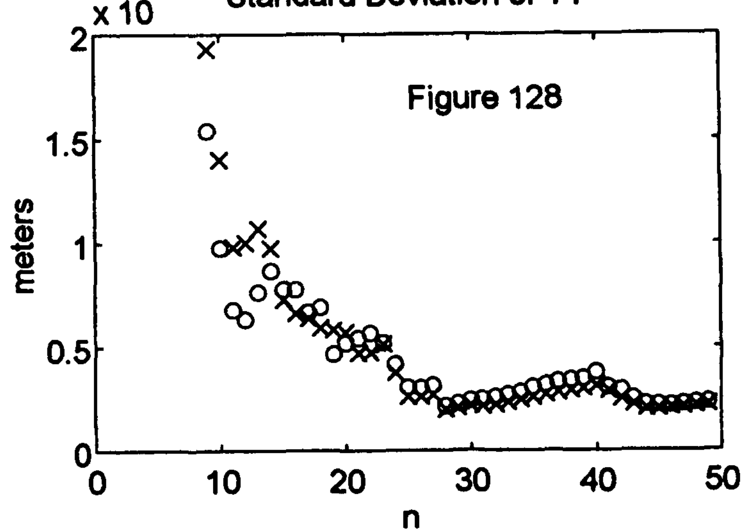

Magnitude of Position Error (Xo,Yo)
Figure 124
meters
500
400
300
200
100
0
0
10
20
30
40
50
n


Projected Current Position XT
Figure 125
XT (meters)
2000
1500
1000
500
0
-500
0
10
20
30
40
50
n

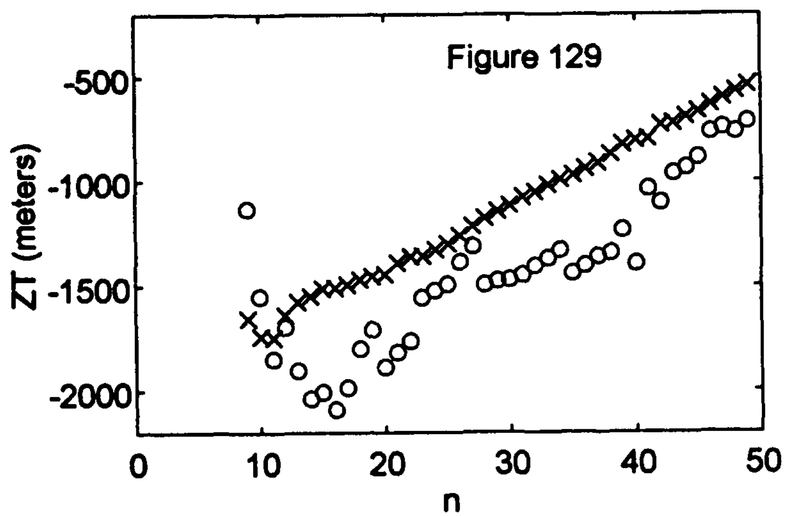

Standard Deviation of XT
Figure 126
x 10^4
meters
2
1.5
1
0.5
0
0
10
20
30
40
50
n



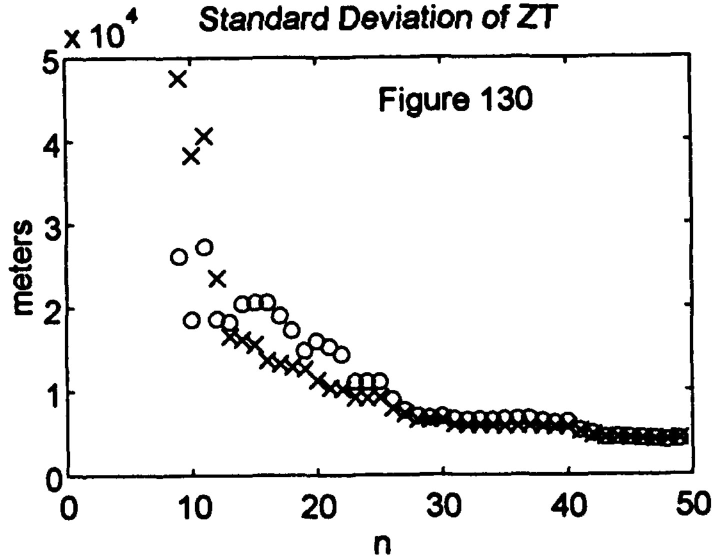
Projected Current Position YT



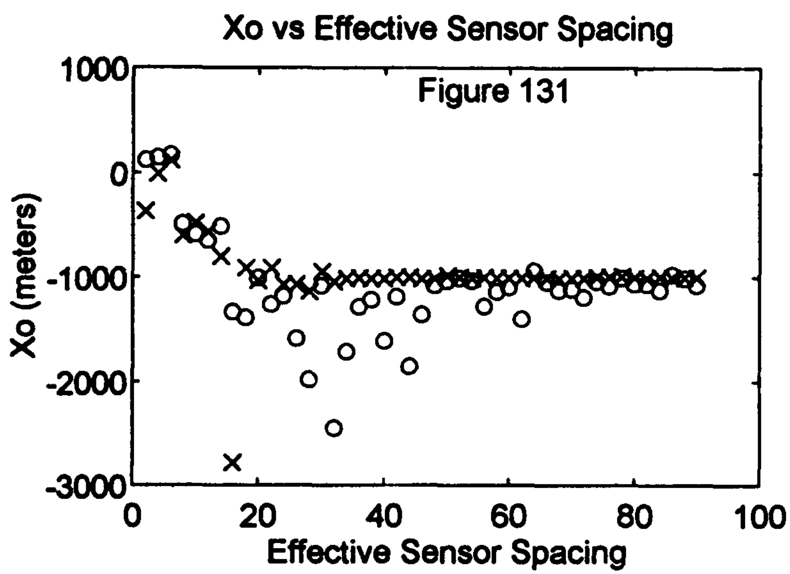
Standard Deviation of YT

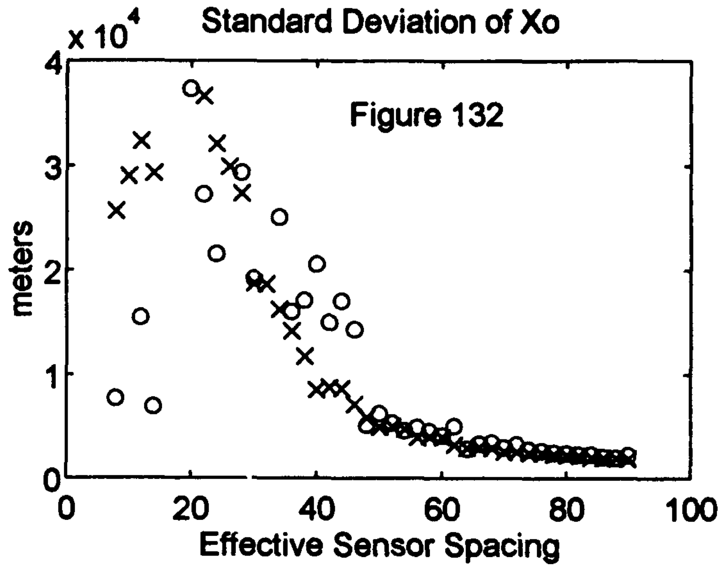


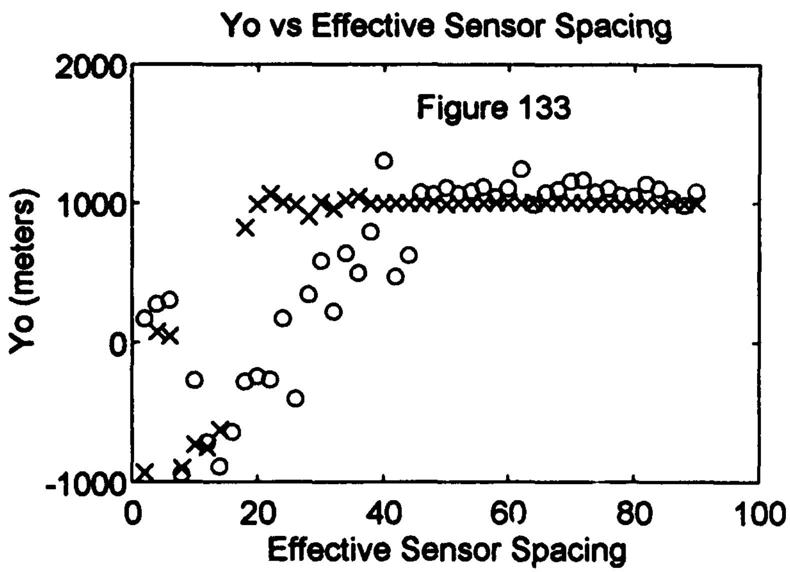
Projected Current Position ZT

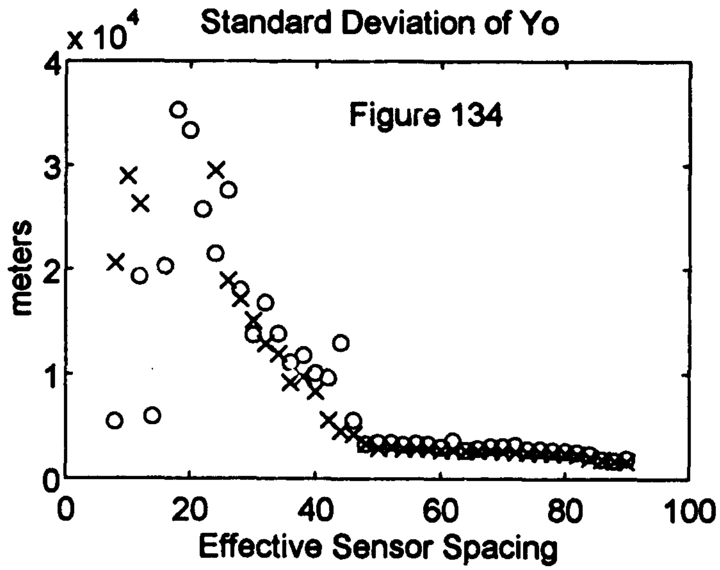


Standard Deviation of ZT

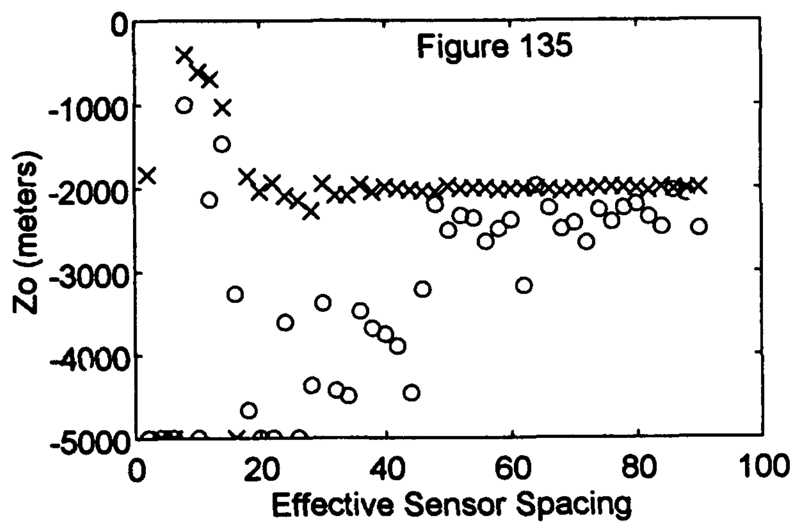

Xo vs Effective Sensor Spacing
Figure 131
Xo (meters)
1000
0
-1000
-2000
-3000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of Xo
x 10^4
Figure 132
meters
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

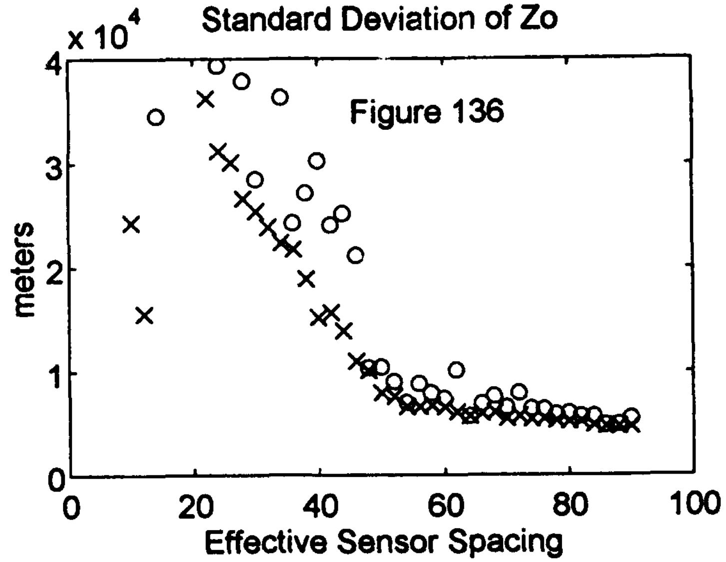

Yo vs Effective Sensor Spacing
Figure 133
Yo (meters)
2000
1000
0
-1000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of Yo
x 10^4
Figure 134
meters
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

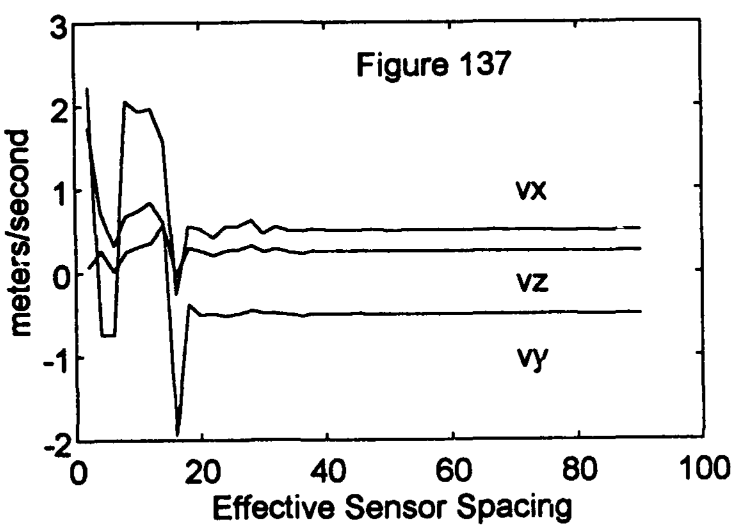
Zo vs Effective Sensor Spacing



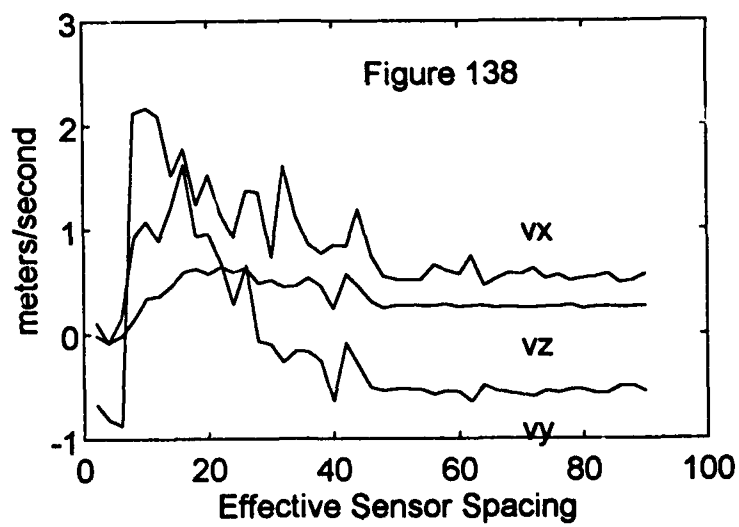
Standard Deviation of Zo



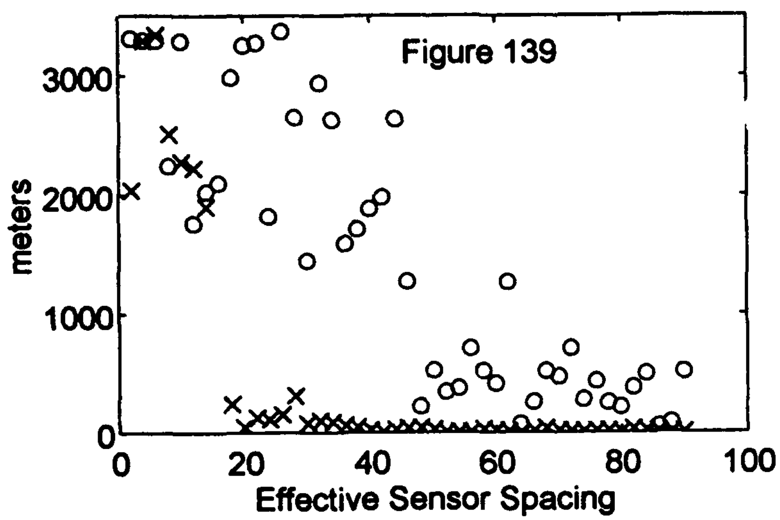
Estimated Velocities 3m GPS Errors



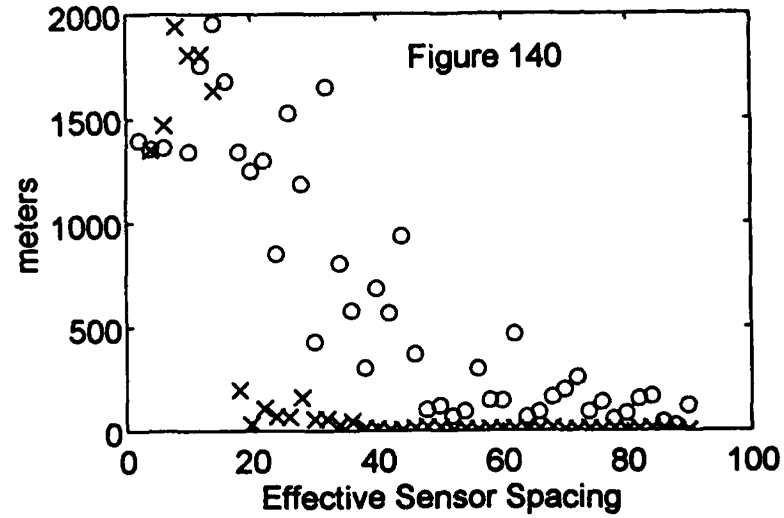
Estimated Velocities 25m GPS Errors



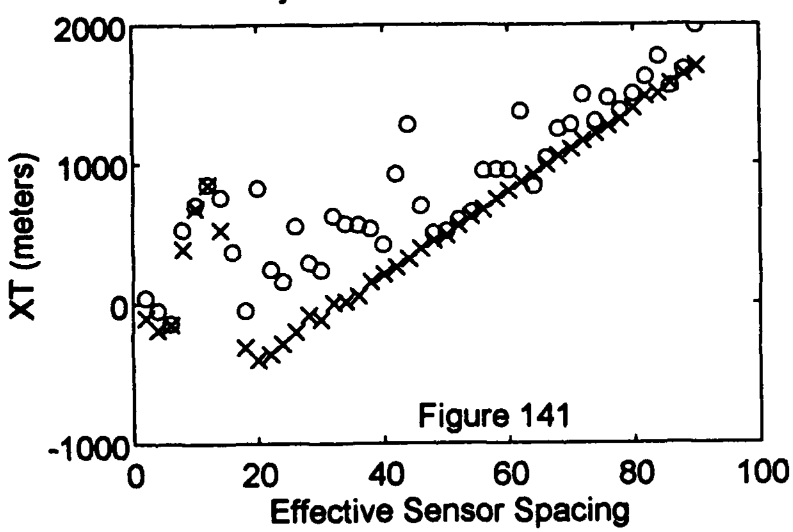
Magnitude of Position Error (Xo,Yo,Zo)



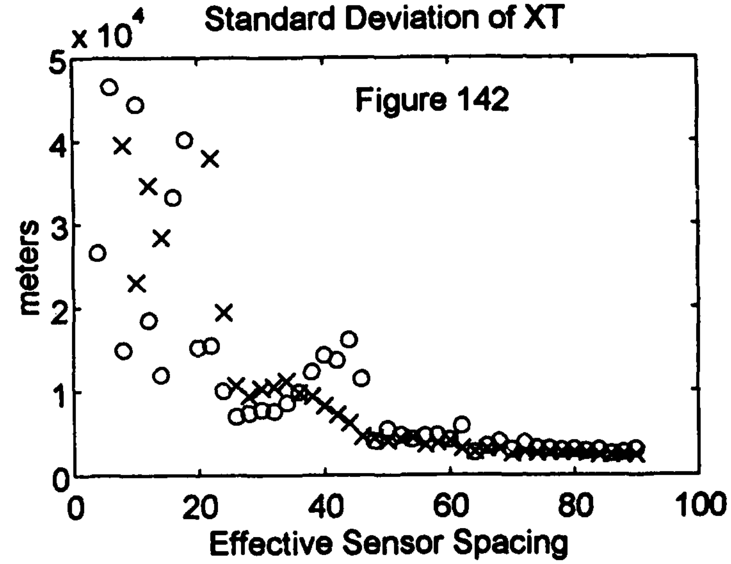
Magnitude of Position Error (Xo,Yo)

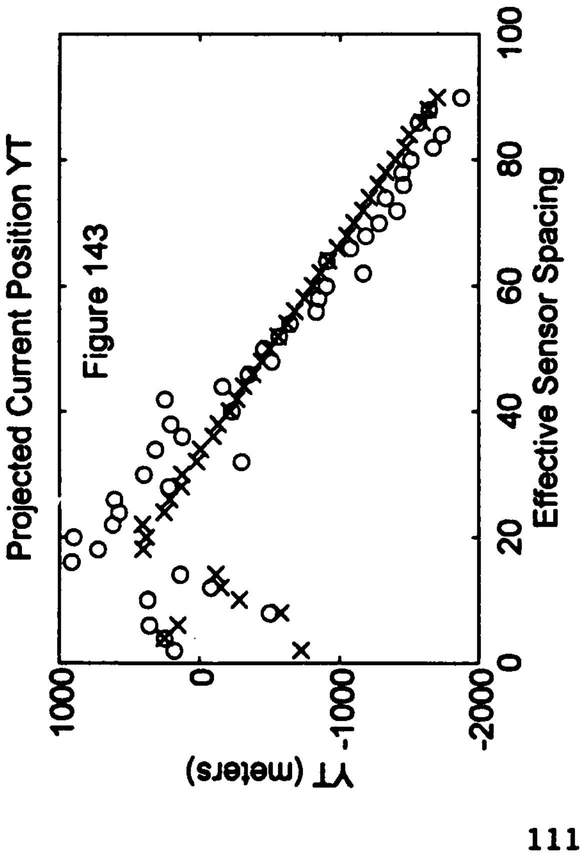


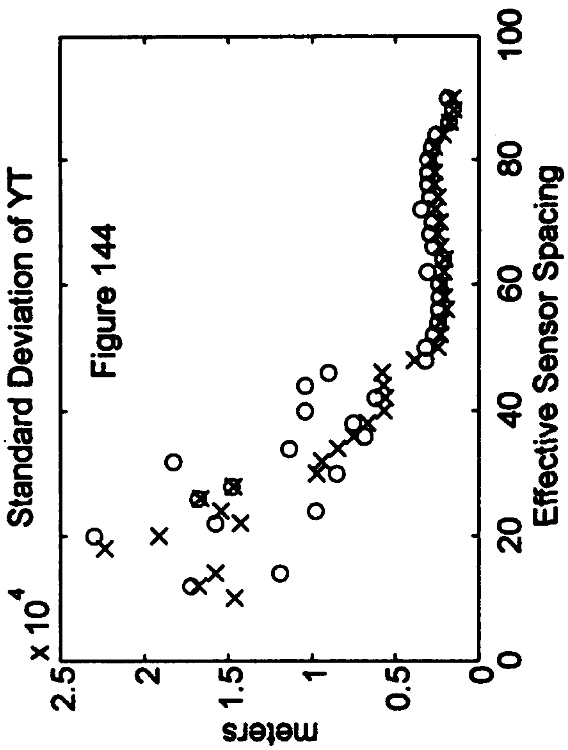
Projected Current Position XT

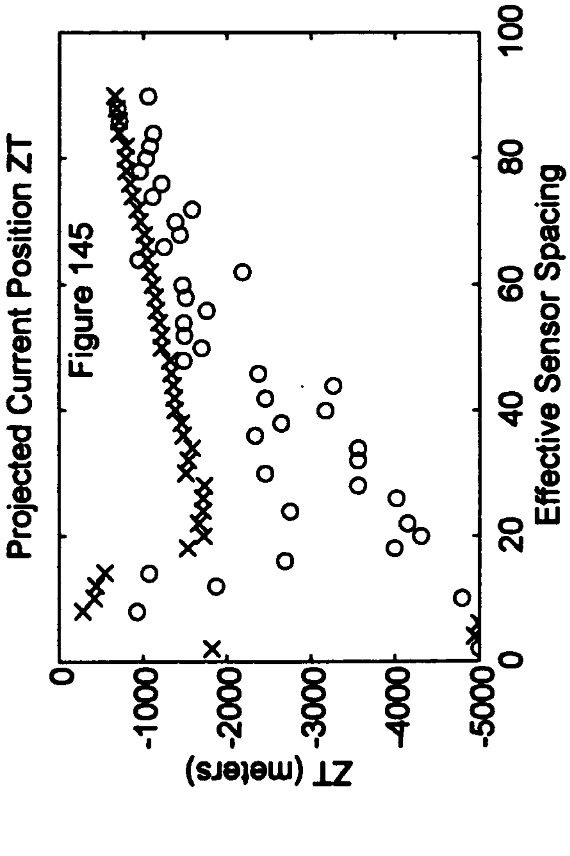


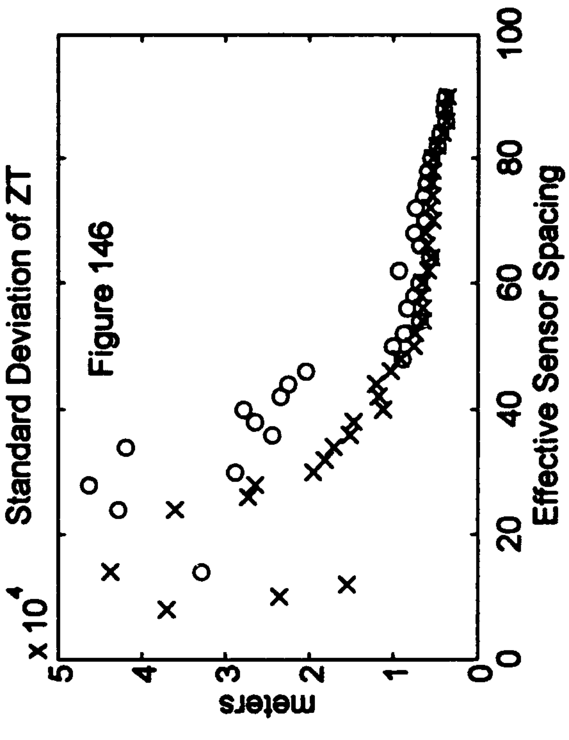
Standard Deviation of XT




Projected Current Position YT
Figure 143
YT (meters)
1000
0
-1000
-2000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of YT
Figure 144
x 10^4
meters
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
Effective Sensor Spacing


Projected Current Position ZT
Figure 145
ZT (meters)
0
-1000
-2000
-3000
-4000
-5000
0
20
40
60
80
100
Effective Sensor Spacing


Standard Deviation of ZT
Figure 146
x 10^4
meters
5
4
3
2
1
0
0
20
40
60
80
100
Effective Sensor Spacing

4.4.4 Moving Acoustic Beacon Summary

From scenarios 4 through 6 we showed that as the number of experiment vectors used in the maximum likelihood estimation problem increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, while the standard deviation of the estimated position decreases. Also, as the effective sensor spacing increases, the accuracy of the estimated position and velocities of the acoustic beacon improves, and the standard deviation of the estimated position decreases. For all of the moving acoustic beacon scenarios, we see that the depth of the acoustic beacon is the least accurately estimated parameter. This is because we are still attempting to locate the acoustic beacon in three dimensions, while the receiving ship only moves in two dimensions. We also note that the standard deviations of the estimated beacon position are significantly larger for the moving beacon scenarios then for the stationary beacon scenarios. These larger standard deviations are due to the nature of the structural model and reflect the difficulties in locating a moving acoustic beacon using only a single omnidirectional hydrophone as a receiving source.

Similar to the stationary acoustic beacon scenarios, to obtain the best possible estimate of the acoustic beacon's location we need to use a large number of experiment vectors with a large effective sensor spacing. Again we run into problems with the time it takes to collect the required measurement vectors. For example, with an effective sensor spacing of 60 pulse repetition intervals, a pulse repetition interval of 2 seconds, and using 50 experiment vectors it takes 100 minutes to collect the required measurement vectors. As with the stationary acoustic beacon scenarios, if time is an

issue, we must reduce the number of experiment vectors used, and reduce the effective sensor spacing. As a minimum, we must use at least 30 experiment vectors and a minimum effective sensor spacing of 45 pulse repetition intervals. With these values, for the search path of figure 4-1 and for a pulse repetition interval of 2 seconds, we should be able to obtain a decent estimate of the acoustic beacon's position and velocities.

4.5 Summary

From the results of the Monte Carlo simulations we see that in general, as the number of experiment vectors used to find the estimated parameter vector θ increases, the errors of the estimated parameters and the variance of the estimated parameter vector deceases. This implied that to obtain the best estimate of the acoustic beacon's location, we should use a large number of experiment vectors, with a large effective sensor spacing. We run into problems with this is approach if the time it takes to obtain an estimated position is a factor. We showed that for the search path given in section 4.2 and for an acoustic beacon with a pulse repetition interval of 2 seconds, we need to use a minimum of 20 experiment vectors and an effective sensor spacing of 35 pulse repetition intervals to obtain a decent estimate when using the stationary acoustic beacon algorithm. We also showed that with the same search path and pulse repetition interval, we need to use a minimum of 30 experiment vectors and an effective sensor spacing of 45 pulse repetition intervals. In all scenarios, the use of accurate navigational equipment to measure the position of the receiving ship is critical to obtain an accurate estimate of the acoustic beacon's location. This is shown by the

better estimates obtained from using Differential/P code GPS errors of 3 meters rms.

Chapter V

Receiver Design

5.1 Introduction

A receiver must be designed to implement the maximum likelihood estimation algorithms described in chapters II and III. This receiver should fulfill the goals of being an inexpensive, lightweight, easily transported receiver that can perform all of the functions of an older chart recorder. To determine the specifications for the receiver hardware, we look at the signal that is received, the ambient noise that is received, the thermal noise generated by the receiver's preamplifier, and the required signal output. We also examine the characteristics of the hydrophone used as a receiving sensor, and develop a practical hydrophone model so the preamplifier's input characteristics can be determined.

In designing a receiver to implement the algorithms described in chapters II and III, we make the following general assumptions. We assume that the beacon signal has a carrier frequency of 10,000 Hertz, a pulse width of 10 milliseconds, and a pulse repetition interval of 2 seconds. This is similar to several acoustic beacons currently in use. To cut costs, a Motorola 68HC11 microcontroller will be used. The 68HC11 is an 8-bit, low power microcontroller with several useful peripheral functions built in. These peripheral functions include an eight channel 8-bit analog to digital (A/D) converter, an asynchronous serial communications interface, five general input/output ports, and a 16-bit free running timer system with five output compare registers. This lets us use a single chip to synchronize the time of receipt t_μ to the

pulse repetition interval, digitally sample the received signal, control the receiver's gain, and communicate with the interfacing computer. Another factor in deciding to use the 68HC11 microcontroller is that the 68HC11 and it's programming support equipment are commonly available. For the receiving sensor, we use the Benthos AQ-4 hydrophone attached to 100 meters of RG-58/u coaxial cable. The Benthos AQ-4 hydrophone has an open circuit response of -201 dB re 1 Volt per 1μPa, which is representative of the type of omnidirectional hydrophones currently available.

5.2 Design Criterion

The signal transmitted by the acoustic beacon can be characterized as a sequence of gated continuous wave pulses at a carrier frequency f_c. We depict the beacon signal as:

$$s_n(t) = \sum_{i=0}^{\infty} w(t - iT_p - T_o)\cdot\cos[2\pi f_c(t - iT_p - T_o) + \phi] \tag{5-1}$$

where w(t) is a windowing function that defines the pulse width, T_p is the period which defines the pulse repetition interval, and ϕ is a phase term. As before, the coefficient T_o represents the unknown absolute time reference.

We can represent the signal that is received by the observer as:

$$R_n(t) = \sum_{i=0}^{\infty} h_i\cdot w(t - iT_p - t_{s\mu})\cdot\cos[2\pi f_c(t - iT_p - t_{s\mu}) + \phi + \zeta_i] + n(t) \tag{5-2}$$

where h_i represents the attenuation the signal experiences in traveling from the beacon to the observer, and ζ_i represents a random phase distortion due the propagation of

the signal through the ocean. As discussed in chapter II, the term $t_{s\mu}$ represents the synchronized time of receipt and is given by:

$$t_{s\mu} = t_{\mu} - i \cdot T_p = T_R + T_o \tag{5-3}$$

where T_R *is the acoustic travel time of the beacon signal. The term* $n(t)$ *represents* the combination of the ambient noise that is received and the thermal noise that is generated by the receiver's preamplifier.

Since we use range differences in the maximum likelihood estimation algorithms of chapters II and III, the parameter that we need to find from the received signal is the synchronized time of receipt $t_{s\mu}$. In order to find $t_{s\mu}$, we digitally process the received signal. To determine the proper sampling rate, we must look at the frequency content of the received signal and the desired accuracy of the synchronized time of receipt $t_{s\mu}$. If we apply the Nyquist sampling rate theorem directly to the received signal, the sampling rate would have to be greater then twice the carrier frequency f_c. For carrier frequencies around 10,000 Hertz, the required sampling rate is too fast for the 68HC11 to handle both the data collection and the signal processing. However, from looking at equation (5-2), we notice that the received windowing function w(t) contains the parameter $t_{s\mu}$. Therefore, if we remove the carrier frequency and just look at the envelope of the received signal given by w(t), we can still determine the synchronized time of receipt by observing when the signal's envelope is received. To remove the carrier frequency f_c, we use a precision rectifying circuit with a low pass filter to form an envelope detector. This allows us sample at the reduced rate of twice the bandwidth of the signal's envelope given by w(t).

To reduce the receiver's overall noise level, the bandwidth of the receiver should match the bandwidth of the envelope. If the bandwidth of the receiver increases above the bandwidth of the received signal's envelope, out of band noise from the oceanic environment and other interference sources enters the receiver, decreasing the signal to noise ratio. On the other hand, if the receiver's bandwidth is less then the bandwidth of the received signal's envelope, we do not allow the full bandwidth of the envelope to pass through the receiver, which again lowers the signal to noise ratio. The windowing function w(t) can be approximated by a simple rectangular window with a duration equal to the pulse width of the signal. The bandwidth of the windowing function and the receiver is then given by:

$$Bandwidth = \frac{1}{PulseWidth} \tag{5-4}$$

For a pulse width of 10 milliseconds, the windowing function's bandwidth is 100 Hertz. With a bandwidth of 100 Hertz, the sampling frequency must then be at least 200 Hertz, which equates to a sampling interval of no more then 5 milliseconds.

We must be careful not to sample at too slow a rate, otherwise the errors associated with the quantization of the time base will cause large errors in the localization algorithms. To determine an acceptable quantization step size, or equivalently, how much error can we tolerate in the time of receipt data, we need to compare the quantization errors to the errors in measuring the observer's location. The source of error in measuring the observers position is due to errors in the GPS position data. For a C/A-code GPS receiver, the published rms errors are on the order of 25

meters. Using a simple homogeneous, direct path propagation model with the speed of sound in water equal to 1500 meters per second, a signal takes 16.667 milliseconds to travel 25 meters. We call this the equivalent time error of the position error. We would like the errors associated with the quantization of the time base to be less than the equivalent time error of the position errors. Assuming no other errors in the time of receipt, the maximum error in measuring the time of arrival $t_{s\mu}$ is half the sampling interval. For a C/A code GPS receiver the required sampling interval of 5 milliseconds is much less then the equivalent time error of the position error, therefore the quantization errors are relatively small compared to the errors in measuring the observer's position. However, because more accurate navigational equipment will most likely be available in the near future, and because the 68HC11 has the required speed, we will sample the received signal's envelope every 1 millisecond. With a sampling interval of 1 millisecond, as long as the rms errors of the navigational system are greater than 1.5 meters, the quantization errors will be smaller then the equivalent time errors of the position errors.

To determine the synchronized time of receipt $t_{s\mu}$ from the sampled envelope, we pass the sampled envelope through a matched filter based upon w(t), and look for the time that corresponds to the maximum value of the output of the matched filter. The received time is then divided modulo the pulse repetition interval to obtain the synchronized time of receipt. The use of a matched filter has two benefits. First, the output of a matched filter for a rectangular input signal with the same pulse width of w(t) will have a unique maximum peak which we can use to calculate the time of

receipt. Secondly, a matched filter produces an output with the maximum signal to noise ratio. The digital form of a matched filter for a rectangular signal has the form of the moving average:

$$y[n] = \frac{1}{N+1} \sum_{i=0}^{N} x[n-i] \tag{5-5}$$

For a pulse width of 10 milliseconds and a sampling rate of 1 millisecond, N is equal to 10. Figures 5-1 and 5-2 show the output of the moving average matched filter for an input that represents the sampled envelope of the received signal with additive noise. From figure 5-2 we see that it is relatively easy to find the maximum value of the output of the matched filter. We do not have to worry about the phase delay of the matched filter because we compare the time of arrival of two signals relative to each other. Since the phase delay is the same for all received signals, it does not effect the comparison of the two arrival times.

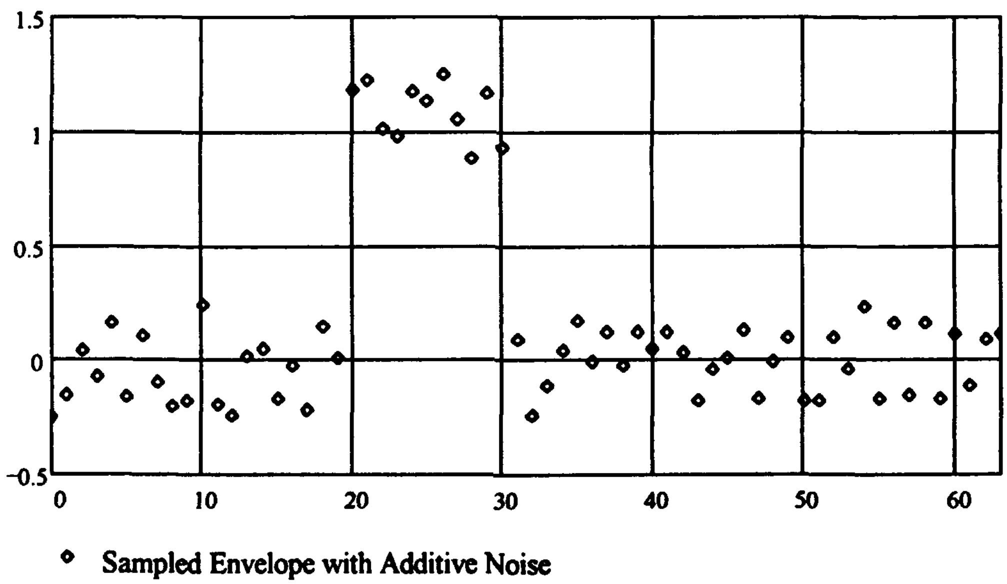


Figure 5-1

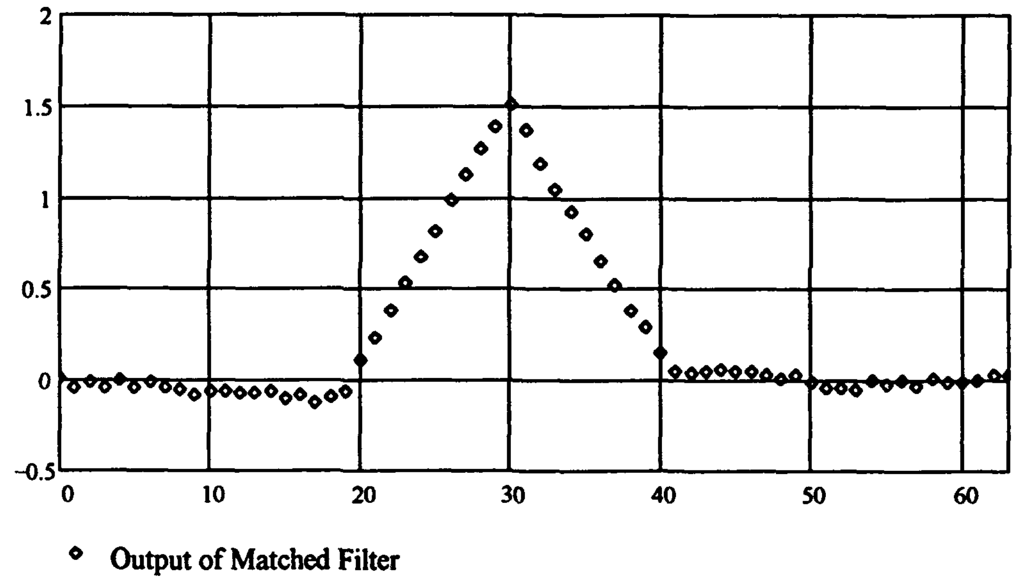


Figure 5-2

The receiver should be able to detect the target's beacon signal in the presence of the additive noise term $n(t)$. The two largest contributors of noise are ambient oceanic noise and thermal noise generated by the receiver's preamplifier. Ambient oceanic noise, due to factors such as wind force, ships, industrial activity, precipitation, biologics, and others, is received by the hydrophone which generates an equivalent noise voltage. Using the hydrophone's open circuit response characteristics, the receiver's bandwidth, and empirical data of noise spectrum levels summarized by Wenz [Wenz 1962], the equivalent noise voltage of the oceanic noise can be calculated using:

$$e_n = 10^{\left(\frac{H_{oc} + e_o}{20} + \log(BW)\right)} \tag{5-6}$$

where H_{oc} is the hydrophones open circuit response in dB re 1 Volt per 1µPa, e_o is the

ambient oceanic noise in dB re 1μPa, and BW is the receiver's bandwidth in Hertz. Table 5-1 shows the equivalent noise voltage for various Sea states (Beaufort Scale) at 10,000 Hertz using a hydrophone with an open circuit receiver response of -201 dB re 1 Volt per 1μPa, and a receiver bandwidth of 100 Hertz. Ideally the noise in a receiver should be dominated by external noise sources. This means that the receiver's thermal noise should be less then the ambient oceanic noise. Since sea states 1 and 2 are not common, the receiver's thermal noise is designed to be less then the ambient oceanic noise associated with sea state 3. This requirement limits the test receiver's thermal noise reflected at the input of the preamplifier to be less then 89 $nV/\sqrt{Hz}$.

Sea State (Beaufort Scale)	Ambient Oceanic Noise (dB re 1μPa)	Equivalent Noise Voltage ($nV/\sqrt{Hz}$)	Total Equivalent Noise Voltage e_n (μV)
1	48	22	0.22
2	56	56	0.56
3	60	89	0.89
5	68	224	2.24
8	73	398	3.98
limit	80	891	8.91

Table 5-1

Another desirable characteristic of the receiver is that it should have as large a dynamic resolution as possible. A large dynamic resolution allows the receiver to detect a broad level of signal amplitudes for a set gain. Since the received signal will be digitally processed, we need to look at the dynamic resolution of the A/D converter.

The dynamic resolution of an A/D converter is given by:

$$\textit{Dynamic Resolution} = 20\ \log_{10}(2^{bits}) \tag{5-7}$$

For and 8 bit A/D converter, the dynamic resolution is only

$$20\ \log_{10}(2^{8}) = 48\ dB \tag{5-8}$$

This is not quite large enough. We would like a dynamic resolution that is comparable to the dynamic resolution of the older paper recorders. This means we need a dynamic resolution of at least 60 dB. To increase the dynamic resolution, either a higher bit A/D converter could be used, or a logarithmic compressor can be used prior to digital sampling. Since we are restricted to the 8 bit A/D converter of the 68HC11, a logarithmic compressor will be used to increase the receiver's dynamic resolution.

Using a logarithmic compressor, the dynamic resolution is given by:

$$\textit{Dynamic Resolution} = 20 \cdot \log_{10}\left(\frac{V_{sat}}{V_{min}}\right) \tag{5-9}$$

where V_{sat} is the saturation point of the logarithmic compressor and V_{min} is the minimum detectable signal at the input of the logarithmic compressor. Typically, logarithmic compressors require an input signal that is greater then 2 mV. Below 2 mV, the input current to the logarithmic compressor becomes comparable to the bias current of the matched transistors used to obtain the logarithmic dependence, and

subsequently the logarithmic dependence is no longer valid [Millman 1979]. For the receiver designed in section 5.4, V_{sat} is approximately 8 V which gives a dynamic resolution of:

$$\textit{Dynamic Resolution} = 20 \cdot \log_{10}\left(\frac{8}{2 \cdot 10^{-3}}\right) = 72\ dB \tag{5-10}$$

To determine the required gain of the receiver, we need to look at the signal level that we are attempting to receive. We wish to be able to detect low level signals that are right at the noise floor of the receiver. This requires the receiver to have enough gain to detect a signal at the equivalent oceanic noise level of 89 $nV/\sqrt{Hz}$. Since a logarithmic compressor is used just prior to A/D conversion, the required gain is determined by the minimum detectable level in the logarithmic compressor and the oceanic noise level. Using the ambient oceanic noise and the input requirements of the logarithmic compressor from above, the maximum required gain of the receiver is given by:

$$A_{v_{max}} = \frac{2 \cdot 10^{-3}}{89 \cdot 10^{-9}} = 87\ dB \tag{5-11}$$

However, it is possible that the received signal could be quite a bit larger then the minimum signal level for a strong beacon in close proximity to the receiving ship. For example, a 30 Watt beacon 100 meters from the receiving hydrophone will produce an input signal of about 10 mV. This signal would completely saturate the receiver if the gain is fixed at $A_{V_{max}}$. To prevent the receiver from saturating, some form of feedback gain control is needed. Using one of the output ports of the 68HC11

as a control line, simple resistive attenuators can be switched on or off to reduce or increase the gain of the receiver based upon the maximum signal level at the A/D converter. To determine how much attenuation is needed, we look at the saturation point of the preamplifier and the maximum likely input signal. For design purposes, to keep a 10 mV signal from saturating the receiver, the receiver's minimum gain is given by:

$$A_{V_{min}} = \frac{V_{sat}}{10 \cdot 10^{-3}} = \frac{8}{10 \cdot 10^{-3}} = 58\, dB \tag{5-12}$$

Here again V_{sat} is the logarithmic compressor's saturation level. Since the input signal level will be somewhere between the noise floor and the maximum likely input signal most of the time, we can use multiple resistive attenuator networks to set the receiver's gain to any level in between $A_{V_{max}}$ and $A_{V_{min}}$. For this design we will use four resistive attenuator networks to provide attenuation levels of -3 dB, -9 dB, -12 dB, -18 dB, -24 dB, and -30 dB.

5.3 Hydrophone Model

To determine the input characteristics of the preamplifier, the hydrophone and cable must be modeled. Figure 5-3 shows an equivalent circuit for a stiffness-controlled piezoelectric hydrophone. Here R_h is the hydrophone's resistance, C_h is the hydrophone's capacitance, e_n represents the equivalent ambient noise of the ocean, and e_{hn} represents the thermal noise of the hydrophone. For frequencies considerably below 100 kHz, the ambient noise level of the ocean is much greater then the

hydrophone's thermal noise, while the impedance of the hydrophone is dominated by the capacitance C_h. This allows us to simplify the hydrophone model by eliminating R_h and e_{hn} [Wilson 1985].

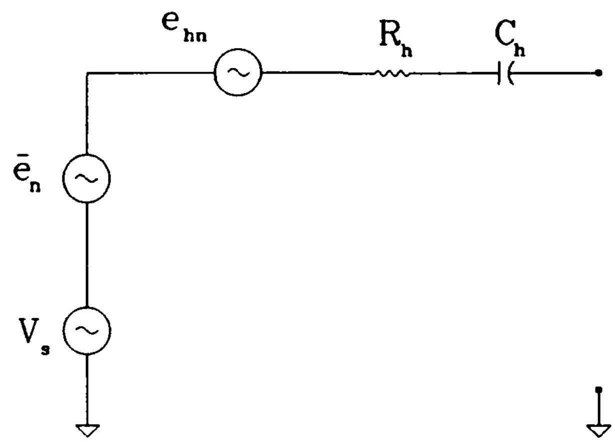


Figure 5-3

The cable assembly can be modeled as a two conductor transmission line. Using standard RG-58/u *coaxial cable with a solid copper inner* conductor and a polyethylene dielectric, we can model the cable as a series resistance R_c and a shunt capacitance C_c [Cheng 1983]. Standard RG-58/u coaxial cable has a series resistance of 33.31 Ω per 1,000 meters, and shunt capacitance of 70 pF per meter. Combing the simplified hydrophone model and the cable model, we have the circuit of figure 5- 4.

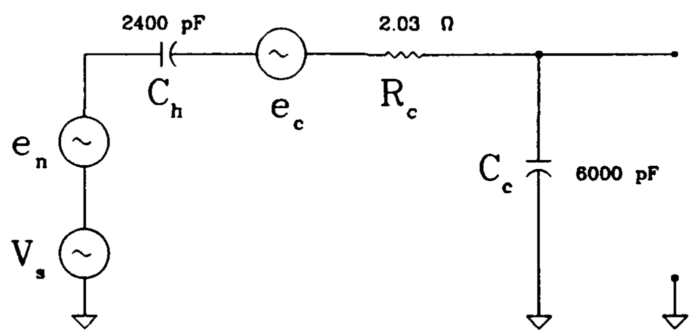


Figure 5-4

Figure 5-4 can be simplified to give the equivalent Thevinin circuit model:

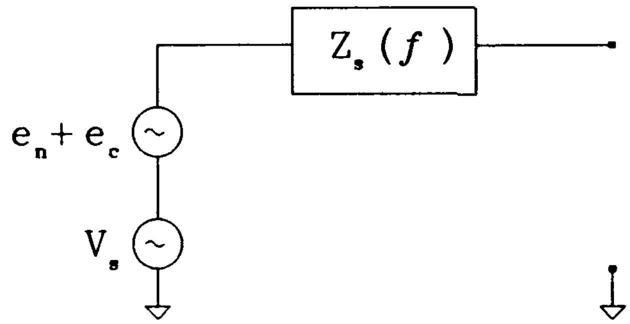


Figure 5-5

The source impedance $Z_s(f)$ is given by:

$$Z_s(f) = \frac{2j\pi f R_c C_h + 1}{\left[2j\pi f R_c C_h C_c + (C_h + C_c)\right] 2j\pi f} \tag{5-13}$$

Figure 5-6 shows a plot of the source impedance verses frequency for the Benthos AQ-4 hydrophone with C_h specified as 2,400 pF ±25%. To prevent serious loading effects, the preamplifier's input impedance must be considerably larger then the source impedance given by equation (5-13). For design purposes, the magnitude of the source impedance at 10,000 Hertz is 1,693 Ω, therefore we choose the preamplifier's input impedance to be at a minimum of 20 time the source impedance, or 33.8 kΩ. This will keep the loss due to loading below 0.5 dB.

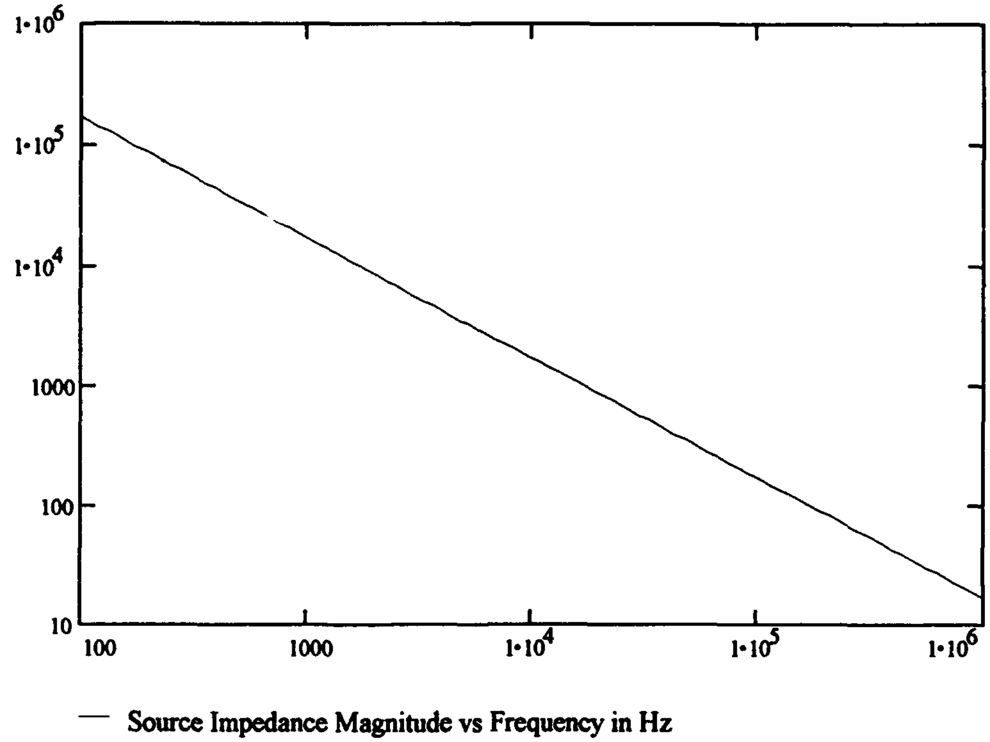


Figure 5-6

5.4 Test Receiver Hardware Design

Based on these design criteria, a receiver was built for testing. This section discuses the circuit design considerations and implementation of this test receiver. Figure 7 shows the general signal flow of the receiver starting with the preamplifier and moving to the 68HC11 microcontroller. An important theme of the receiver design is to avoid exotic components to minimize the cost of the receiver.

5.4.1 Preamplifier

Refer to the preamplifier schematic diagrams in Appendix 1 for the following description. The first stage of the preamplifier is a tuned N channel junction field effect transistor (JFET) common-source amplifier. To help keep low frequency ambient oceanic noise from entering the preamplifier, C_{01} and R_{01} form a high pass

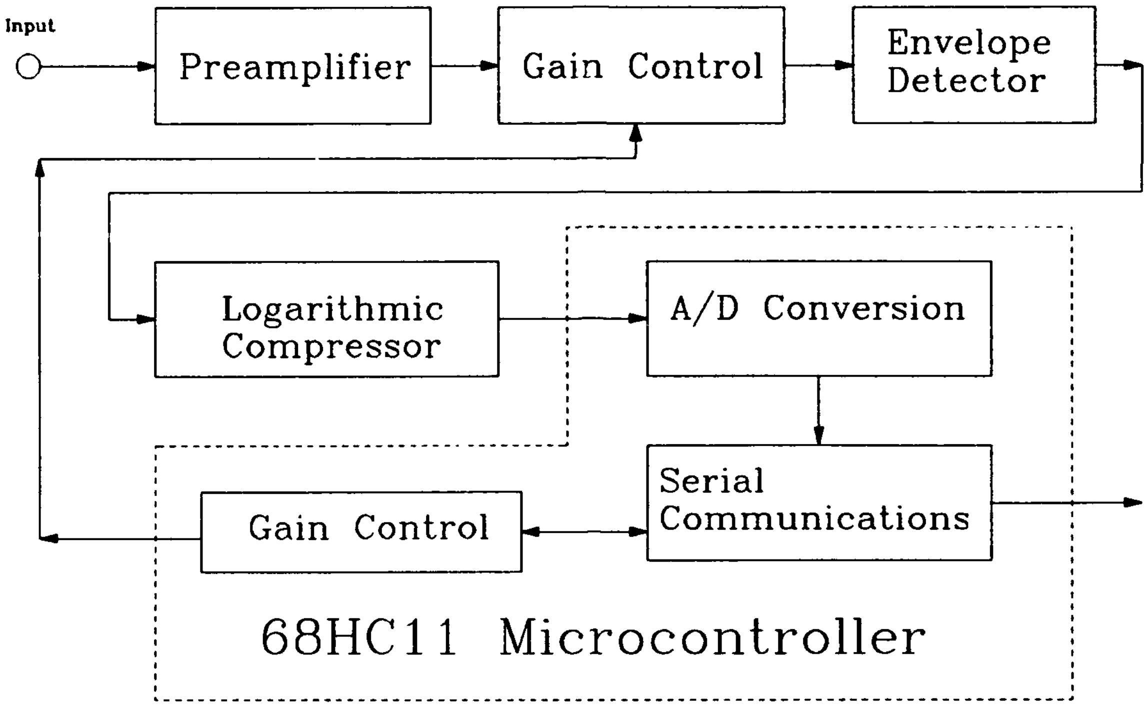


Figure 5-7

filter with a cutoff frequency given by:

$$f_c = \frac{1}{2\,\pi\,R_{01}\,C_{01}} \tag{5-14}$$

To eliminate as much low frequency oceanic noise as possible but still allow for small changes in the acoustic beacon's carrier frequency, we chose the high pass filter cut off frequency as 7000 Hertz. The preamplifier's input impedance is determined by R_{01} since the input impedance of the JFET is so large ($>10^{14}\ \Omega$). To prevent loading between the hydrophone source and the preamplifier we must ensure R_{01} is larger then 33.8 kΩ. To ensure that the thermal noise of the preamplifier is less then equivalent oceanic noise of sea state 3, R_{01} must be chosen such that the thermal noise of the parallel combination of R_{01} and source impedance $Z_s(f)$ is considerably less then

89 nV/√Hz. The thermal noise of a resistor is given by:

$$\bar{e}_n = \sqrt{4ktRB} \tag{5-15}$$

where k is Boltzmann's constant, t is the absolute temperature in Kelvin, B is the bandwidth in Hertz, and R is the resistance in Ohms [Horowitz and Hill 1989]. For the Benthos AQ-4 hydrophone at a frequency of 10,000 Hertz $Z_s(f)$ is approximately 1700 Ω. As a result, the parallel combination of R_{01} and $Z_s(f)$ will be close to the value of $Z_s(f)$ for R_{01} greater then 32 kΩ, and the associated thermal noise will be well below 89 nV/√Hz. However, it would be nice to ensure that the thermal noise of the preamplifier is much less then equivalent oceanic noise of sea state 3 for any source impedance. To ensure this, we choose R_{01} such that its thermal noise is approximately 45 nV/√Hz. To meet the noise constraint, the loading constraint, and the cut off frequency constrain of equation (14), we choose R_{01} as 120 kΩ and C_{01} as 180 pF.

The gain of the first stage JFET common-source amplifier is given by:

$$A_{v_1}(f) = \frac{-Y_{fs} \cdot Z_D(f) \cdot r_d}{Z_D(f) + r_d} \tag{5-16}$$

where Y_{fs} is the JFET's forward transconductance, r_d is the JFET's intrinsic drain resistance, and $Z_D(f)$ is given by:

$$Z_d(f) = \frac{\dfrac{2j\pi f}{C_{02}}}{(j\pi f)^2 + \dfrac{2j\pi f}{R_{02}C_{02}} + \dfrac{1}{L_{01}C_{02}}} \tag{5-17}$$

$Z_D(f)$ represents the complex impedance of the combination of L_{01}, R_{02}, and C_{02}. The use of the reactive load $Z_D(f)$ allows us to obtain the narrow bandwidth required by equation (5-4). The gain of the first stage will be greatest when the frequency of the input signal is at the resonant frequency of $Z_D(f)$. The resonant frequency of $Z_D(f)$ is given by:

$$f_r = \frac{1}{2\pi\sqrt{L_{01} C_{02}}} \tag{5-18}$$

We want the preamplifier to have it's maximum gain at the acoustic beacon's carrier frequency, therefore we choose L_{01} and C_{02} such that the resonant frequency equals the acoustic beacon's carrier frequency . In the test receiver, Q_{01} is an E230 general purpose N channel JFET. The E230 has a nominal forward transconductance of $1.5 \cdot 10^{-3}$ Siemens and a nominal intrinsic drain resistance of 12.5 kΩ. Using equations (5-16) and (5-17), and the values from Appendix 1, at a beacon frequency of 10,000 Hz, A_{V1} is approximately 24 dB.

In general, the output impedance of an R-L-C tuned amplifier is quite high, therefore, a JFET source follower is used as a buffer immediately following the first stage. Resistor R_{04} and capacitor C_{04} form a high pass filter similar to R_{01} and C_{01}, and serve to capacitively couple the JFET source follower to the tuned JFET amplifier. The cut off frequency of the high pass filter formed by R_{04} and C_{04} is given by equation (5-14) with R_{04} substituted for R_{01} and C_{04} substituted for C_{01}. To prevent loading between the JFET amplifier and the JFET buffer, R_{04} must be significantly larger then the output impedance of the JFET amplifier. However, we must be careful

in choosing R_{04} so that its thermal noise, when reflected back to the input of the first stage, is less then 89 nV/$\sqrt{}$ Hz. Using equation (15), this limits R_{04} to 560 kΩ.

To enhance the performance of the JFET source follower, Q_{03} and R_{05} are added as a bootstrapped load. Because the base-emitter voltage of Q_{03} is approximately constant, the combination of R_{05} and Q_{03} cause a constant source current to flow through JFET Q_{02}. This makes Q_{02}'s V_{GS} approximately constant, which in return reduces nonlinearities [Horowitz and Hill 1989]. This is beneficial because the signal level at this point in the preamplifier is still quite small.

The first stage of the preamplifier is powered by a separate 5 Volt 3-terminal 7805 voltage regulator. We do this to isolate the first stage from subsequent stages, which prevents the feedback of any noise generated by subsequent stages along the main power supply line. This provides Q_{01} with a clean, stable supply voltage which is critical when working with very low level input signals.

The second stage of the preamplifier is another tuned amplifier, this time using a PNP bipolar transistor. Capacitor C_{05} combined with the parallel combination of the bias resistors R_{07} and R_{08} form another high pass filter while also serving to capacitively couple the JFET source follower from the first stage to the second stage. The cutoff frequency of the high pass filter formed by C_{05}, R_{07} and R_{08} is given by:

$$f_c = \frac{1}{2\pi \cdot C_{05} \dfrac{R_{07} \cdot R_{08}}{R_{07} + R_{08}}} \qquad \text{(5-19)}$$

For this stage, we set the high pass filter's cut off frequency at 3000 Hertz. The gain of the second stage is given by:

$$A_{V_2} = \frac{-Z_c(f)}{R_{10}} \tag{5-20}$$

where

$$Z_c(f) = \frac{\dfrac{2j\pi f}{C_{07}}}{(2j\pi f)^2 + \dfrac{2j\pi f}{R_{11}\,C_{07}} + \dfrac{1}{L_{02}\,C_{07}}} \tag{5-21}$$

As in the first stage we set the resonant frequency of $Z_c(f)$ given by:

$$f_r = \frac{1}{2\pi\sqrt{L_{02}\,C_{07}}} \tag{5-22}$$

to the acoustic beacon's carrier frequency. Using equations (5-20) and (5-21) for a beacon frequency of 10,000 Hertz, A_{V2} is approximately 31 db.

Following the PNP tuned amplifier, a simple JFET source follower is used as a buffer. Capacitor C_{08} and resistor R_{12} form yet another high pass filter while also capacitively coupling the output of the PNP amplifier to the input to the JFET follower. The cut off frequency of the high pass filter formed by C_{08} and R_{12} is given by equation (14) with C_{08} substituted for C_{01} and R_{12} substituted for R_{01},and is set to 3,000 Hertz.

Similar to the first stage, the second gain stage is powered by a separate 5 Volt, 3-terminal 7805 voltage regulator. Again we do this to prevent noise from being feedback along the power supply lines.

At the end of the second stage of the preamplifier the maximum gain at the acoustic beacon's carrier frequency is approximately 55 dB. We are under the minimum required gain of 58 dB, therefore we should not have to worry about the first two stages of the preamplifier saturating the logarithmic compressor. However, we still need an additional 32 dB of gain to reach the required maximum gain of 87 dB. Prior to the next gain stage, we include an attenuation network to prevent subsequent stages from saturating.

Following the JFET source follower Q_{05}, R_{14} in combination with shunt resistors R_{15}, R_{16}, R_{17}, and R_{18} form a variable attenuation network which is controlled by the 68HC11 microcontroller. To reduce the gain of the preamplifier, any combination of R_{15} through R_{18} can be switched to ground through two CD4053 analog switches. Once grounded, the shunt resistors form a voltage divider with R_{14}, and subsequently attenuate the signal. Following the shunt resistors is a simple common emitter amplifier that provides the final gain necessary to obtain a maximum of 87 dB. The gain of this stage, including the effects of the attenuator network is given by:

$$A_{V_3} = \frac{-R_{21}}{R_{22}} \cdot \frac{2\pi f R_S C_{09} \cdot \dfrac{R_{19} R_{20}}{R_{19} + R_{20}}}{\left[\left(R_{14} + \dfrac{R_{19} R_{20}}{R_{19} + R_{20}}\right) \cdot R_s + \dfrac{R_{19} R_{20}}{R_{19} + R_{20}} \cdot R_{14}\right] \cdot 2\pi f C_{09} + R_s + R_{14}} \qquad \text{(5-23)}$$

where R_s is the parallel combination of any shunt resistors R_{15} through R_{18} that are switched to ground. If none of the shunt resistors are switched to ground, R_s is very large and A_{V3} approaches:

$$A_{V_3} = \frac{-R_{21}}{R_{22}} \tag{5-24}$$

In this case the gain of the preamplifier is at its maximum of approximately 89 dB. In actuality, the maximum gain will be slightly less then 89 dB due to small losses caused by loading between the different stages. If all of the shunt resistors are switched to ground, R_s is at its minimum value and the gain of the preamplifier is approximately 56 dB. Table 5-2 summarizes which shunt resistors are switched to ground to obtain different levels of attenuation.

Attenuation Level	Shunt Resistors Switched to Ground
0 dB	none
-3 dB	R_{15}
-9 dB	R_{16}
-12 dB	R_{15} and R_{16}
-18 dB	R_{17}
-24 dB	R_{18}
-30 dB	R_{15}, R_{16}, R_{17}, and R_{18}

Table 5-2

5.4.2 Envelope Detector Circuit

Refer to the Envelope Detector schematic diagram in Appendix 1 for the following description. After the preamplifier, the received signal passes through a unity gain buffer. This isolates the preamplifier from the envelope detector, and also provides the low source impedance that the precision rectifier circuit needs to function properly. U_{07}, Q_{07}, Q_{08}, and the low pass filter combination of C_{25} and R_{31} are the

heart of the envelope detector. Diode D_{01} protects U_{07} during large input swings above the reference voltage V_{REF}. Resistors R_{29} and R_{30} add bias stability to transistors Q_{07} and Q_{08}.

As the input signal at the inverting terminal of U_{07} goes below the reference voltage at the noninverting terminal of U_{07}, Q_{07} is brought into the active region, thereby drawing Q_{08} into the active region which charges capacitor C_{25}. As the difference between the reference voltage and the input signal becomes greater, Q_{07} and Q_{08} are brought further into the active region which in turn increases the time that capacitor C_{25} charges. When the signal at the inverting terminal of U_{07} is above the reference voltage at the noninverting terminal of U_{07}, transistor Q_{07}, and subsequently Q_{08} are cut off. The charge in capacitor C_{25} is then discharged through resistor R_{31}. As long as the time constant formed by the combination of C_{25} and R_{31} is much greater then the period of the input signal's carrier frequency, the output taken at C_{25} will be the envelope of the input signal. To ensure operation with very low level inputs, U_{08} is chosen to have a large gain bandwidth product and a fast slew rate.

5.4.3 Logarithmic Compressor

Refer to the Logarithmic Compressor schematic diagram in Appendix 1 for the following description. Prior to logarithmic compression, we pass the received envelope through a noninverting amplifier with a gain of 9 dB to restore the signal level that was lost in the envelope detector. To obtain an output that is proportional the logarithm of the input, we rely on the logarithmic relation between transistor Q_{09}'s voltage and current. By using Q_{09} in U_{09}'s feedback loop, the output of the logarithmic

compressor is given by:

$$V_o = -\left(1+\frac{R_{41}}{R_{42}}\right)\cdot V_T\cdot\ln\left(\frac{V_i\cdot R_{40}}{R_{35}\cdot V+}\right) \qquad (5\text{-}25)$$

where V_T is given by:

$$V_T = \frac{temperature}{11600} \qquad (5\text{-}26)$$

Here the temperature is in Kelvin. For room temperature, V_T is approximately 0.0259. From equation (5-25) we note that for small input signals, the output V_o is large, and for large input signals, the output V_o is small. To set the zero crossing point of the logarithmic compressor, we choose R_{40} such that:

$$R_{40} = \frac{R_{35}\cdot V+}{V_{sat}} \qquad (5\text{-}27)$$

With R_{35} equal to 10 kΩ, V+ equal to 10 Volts, and V_{sat} equal to 8 Volts, R_{40} is approximately 12.5 kΩ. V_{sat} represents the largest possible input to the logarithmic compressor. To set the maximum output level of the logarithmic compressor, we chose R_{42} and R_{43} such that the minimum input signal of 2 mV produces an output of 5 volts. This allows us to digitally sample the output of the logarithmic compressor directly, without worrying about damaging the 68HC11's A/D converter. The use of the matched transistor Q_{09} helps stabilize the temperature dependence of V_T, and is essential to eliminate the dependence upon the reverse saturation current I_o which doubles for every 10° C rise in temperature [Millman 1979].

By taking the saturation level of the receivers as approximately 8 volts, the

dynamic resolution of the receiver is approximately:

$$\textit{Dynamic Resolution} = 20 \cdot \log_{10}\left(\frac{8}{2 \cdot 10^{-3}}\right) = 72 \ dB \tag{5-28}$$

To obtain the full dynamic resolution at the input to the A/D converter, the output of operational amplifier U_{10} must be capable of going between zero and five volts. We would like to restrict the output of U_{10} to be strictly less then 5.5 Volts to prevent damage to the 68HC11's A/D converter in case of a failure in the logarithmic compressor. To accomplish this, we step down the main positive supply voltage to 5.6 volts using a zener diode reference. The maximum output of the CA3130 operational amplifier used for U_{10} is about 0.5 volts below the positive supply voltage, therefore the output of U_{10} is restricted between 0 and 5.1 volts.

5.4.4 68HC11 Microcontroller

Refer to the 68HC11 Microcontroller schematic diagram in Appendix 1 for the following description. The 68HC11 is run in expanded mode at a clock speed of 8 MHz. Input/Output ports B and C are used to interface to the external 27HC64 EPROM and the 62HC64 RAM chips. A 74HC373 multiplexer and a 74HC00 NAND GATE are used to form the proper address line controls and to transfer data to the external memory chips. Port PE0 is used for the input to the A/D converter. The logarithmically compressed signal passes through a low pass filter to remove any high frequency noise, and then is sequentially sampled by the A/D converter. Ports PD0 and PD1 are used for serial data output. A SCL/U 20 mA SAIL-to-low-power-logic converter is used to convert the 68HC11's serial data to the SAIL communication

protocol. The use of the SAIL communications protocol is beneficial because of it's simplicity, and it's ability to communicate with multiple nodes. The rest of port D's output lines are tied to the positive supply through 100 kΩ pull up resistors to prevent the output lines from floating. Ports PA3, PA4, PA5, and PA7 are used to control the two CD4053 2 channel analog switches which control the attenuator network described earlier. The rest of port A's input/output lines, except PA0, are tied to ground through 100 kΩ resistors to prevent the outputs from floating. PA0 is left as an external input for further development, and is tied to the positive supply through a 100 kΩ pull up resistor.

Appendix 2 contains a listing of the program that the 68HC11 executes.

5.4.5 Power Supply

In the test receiver, a +10 volt power supply is used to power the preamplifier, the envelope detector, and the logarithmic compressor. A secondary +5 volt power supply is used to power the 68HC11 microcontroller. The use of a separate power supply for the microcontroller prevents high frequency noise generated by the digital components from entering the preamplifier. For the -5 V that is required by the logarithmic compressor, a third supply is used. This provides a better regulated negative supply then using a DC to DC voltage converter such as the ICL 7660 to generate the required -5 V. Finally, a +12 volt, 300 mA supply is provided to power an external GPS receiver. All four power supplies are built on a common circuit board and share a common ground bus. Refer to the Power Supply schematic diagram in Appendix 1 for the following circuit description.

The test receiver uses a standard power cord to connect to external power with a one half amp slow-blow fuse to protect the receiver in case of a short circuit. A Magnetek Triad F-93X transformer is used to step down the input voltage. Diodes D_{01} through D_{04} act as a standard full wave bridge rectifier. Voltage regulator U_{01} is a simple three terminal +12 volt regulator used to provide power to the external GPS receiver. Capacitor C_{01} filters the rectified supply voltage to provide an unregulated DC voltage to U_{01}. Capacitors C_{02} and C_{03} are bypass capacitors used to reduce noise on U_{01}'s input and output leads. Because of the high current requirements of the voltage regulator, U_{01} is mounted to the chassis bottom for heat sinking.

Voltage regulator U_{02} is an adjustable four terminal voltage regulator used to provide the +10 volts required by the preamplifier, absolute value circuit, and the logarithmic compressor. Resistor R_{01} and capacitor C_{04} form a low pass filter which helps reduce high frequency feedback from the GPS receiver power supply (U_{01}). R_{01} also acts as an attenuator, reducing the unregulated DC input voltage to regulator U_{02}. By doing this, the power that U_{02} has to sink is reduced, increasing U_{02}'s efficiency. Resistors R_{02}, R_{03}, and R_{04} form the feedback network used by U_{02} to set the output voltage. By using an adjustable voltage regulator, the positive supply voltage can be set more precisely, which is needed to maintain sensitive bias points in the logarithmic compressor. Capacitors C_{05} and C_{06} are bypass capacitors similar to C_{02} and C_{03}.

Voltage regulator U_{03} is a simple three terminal regulator used to supply the +5 volts required by the 68HC11 and support systems. Resistor R_{05} and capacitor C_{07} form another low pass filter which helps isolate the preamplifier from the 68HC11

microcontroller. As with R_{01}, R_{05} reduces the unregulated DC voltage at the input of U_{03}, reducing the power that U_{03} must sink. Capacitors C_{08} and C_{09} are bypass capacitors similar to C_{02} and C_{03}. The combination of resistor R_{06} and LED D_{05} are used as a power indicator, with D_{05} mounted on the front panel of the chassis next to the power switch.

Finally, voltage regulator U_{04} is a four terminal, adjustable negative voltage regulator that is used to provide the -5 volts required for the logarithmic compressor. Resistor R_{07} and capacitor C_{10} form a low pass filter which converts the rectified input voltage to unregulated DC. Resistors R_{08}, R_{09}, and R_{10} form the feedback network used by U_{04} to set the output voltage at -5 volts. As before, C_{11} and C_{12} are bypass capacitors that help reduce noise the voltage regulator's input and output lines.

5.5 Test Receiver Performance

To test the validity of the receiver designed in section 5.4, experiments were conducted to measure the actual performance of the preamplifier, envelope detector, and logarithmic compressor.

5.5.1 Preamplifier Performance

To test the performance of the preamplifier, experiments were conducted to measure the preamplifier's actual gain, bandwidth, and thermal noise. To measure the gain and bandwidth of the preamplifier, the output of the preamplifier was measured for sinusoidal inputs at different frequencies. The results, along with the predicted values are plotted in figure 5-8. From figure 5-8 we see that the actual gain of the preamplifier at the acoustic beacon's carrier frequency is 86 dB. This is close enough

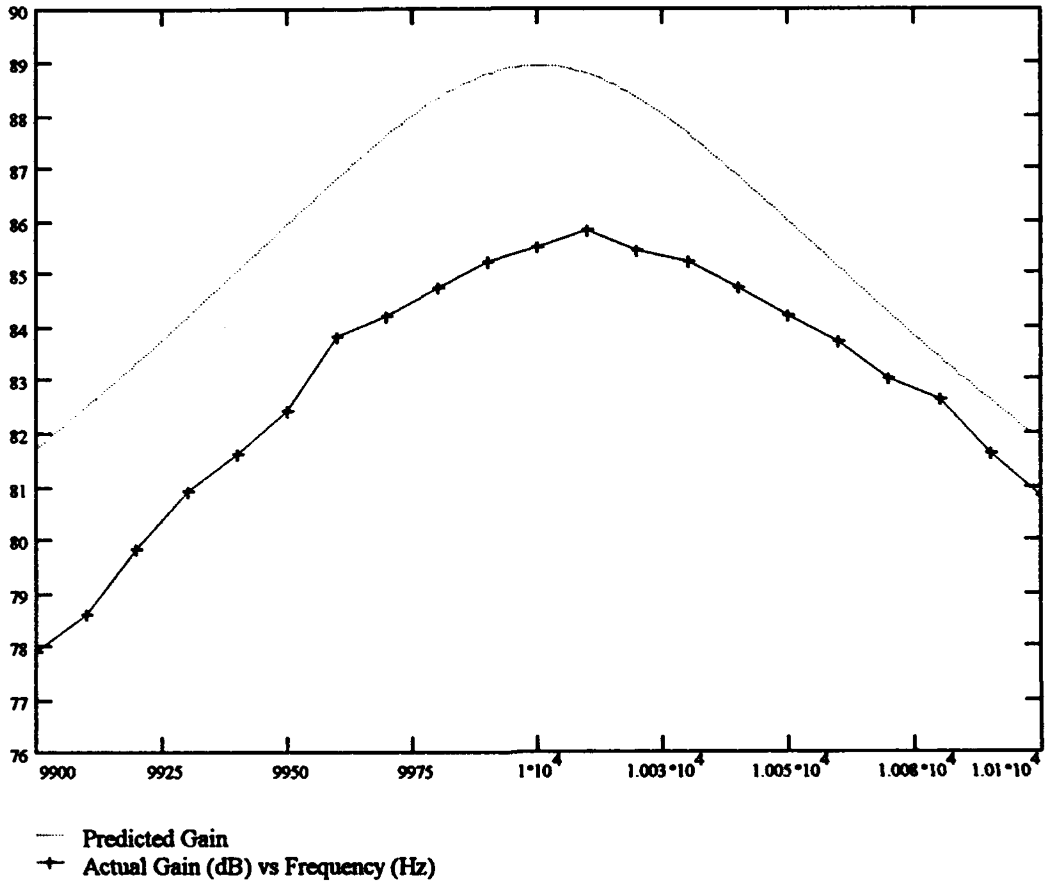


Figure 5-8

to the design goal to be acceptable. The loss in gain compared to the predicted gain is due to the effects of loading between the various stages and to deviations from the nominal values of Y_{fs} and r_d. We also note that the actual bandwidth is 120 Hz which is slightly larger then the predicted bandwidth. This increase is due to deviations of R_{02} and R_{11} from their nominal values and is not significant.

To determine the actual thermal noise of the preamplifier, we short the preamplifier's input and measure the root mean square output using a digital multimeter. Assuming the preamplifier's noise is normally distributed, we correct the

output measurement to reflect a normal distribution by multiplying by 1.13 [Horowitz and Hill 1989]. The preamplifier's thermal noise is then found by using:

$$e_n = \frac{1.13 \cdot e_s}{A_v \sqrt{BW}} \tag{5-29}$$

where e_s is the measured output voltage, A_V is the total gain of the preamplifier, and BW is the bandwidth of the preamplifier. Using this approach, e_s was measured as 6.8 millivolts. The preamplifier's thermal noise is then:

$$e_n \approx \frac{1.13 \cdot 6.8 \cdot 10^{-3}}{19275\sqrt{120}} \approx 31\ nV/\sqrt{Hz} \tag{5-30}$$

To check this value we can observe the output of the preamplifier (with the input grounded) on an oscilloscope. To measure the preamplifier's noise, we take the peak value observed on the oscilloscope. Next we calculate the preamplifier's thermal noise using:

$$e_n = \frac{V_{peak}/6}{A_v \sqrt{BW}} \tag{5-31}$$

This gives a rough approximation of the thermal noise of the preamplifier. V_{peak} was measured as 35 mV. Using equation (5-31), the thermal noise of the preamplifier is roughly 28 nV/√Hz. This value is of the same order of magnitude as found in equation (5-30), so we can assume the noise figure given in equation (5-30) is correct.

5.5.2 Envelope Detector

To test the performance of the envelope detector we look at the output for

different amplitudes of a sinusoidal input. Figure 5-9 shows the measured output of the envelope detector for different amplitudes of a 10,000 Hertz sinusoid input.

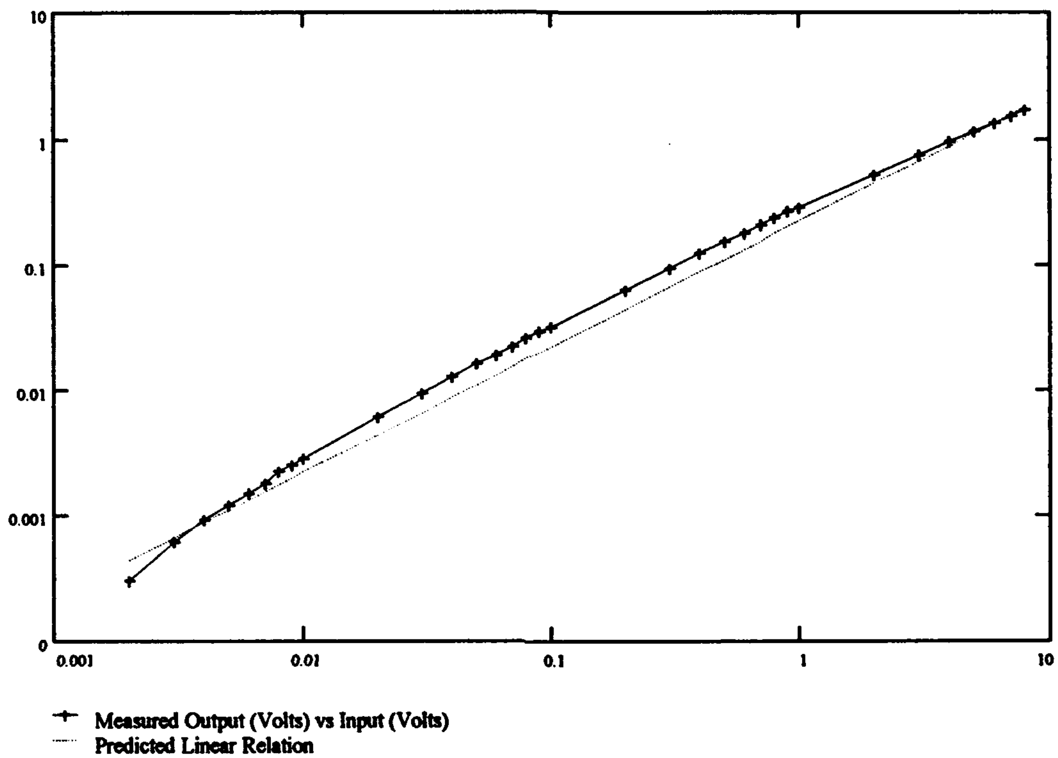


Figure 5-9

Ideally, the envelope detector should have an output that is linearly dependent on the amplitude of the input sinusoid. This allows the envelope detector to accurately track the true envelope of the beacon signal. From figure 5-9 we see that the measured output is nearly linear, therefore we expect the envelope detector to work properly.

5.5.3 Logarithmic Compressor

To test the performance of the logarithmic compressor, we look at the output for different DC voltage inputs. Figure 5-10 shows the measured output values and the predicted values.

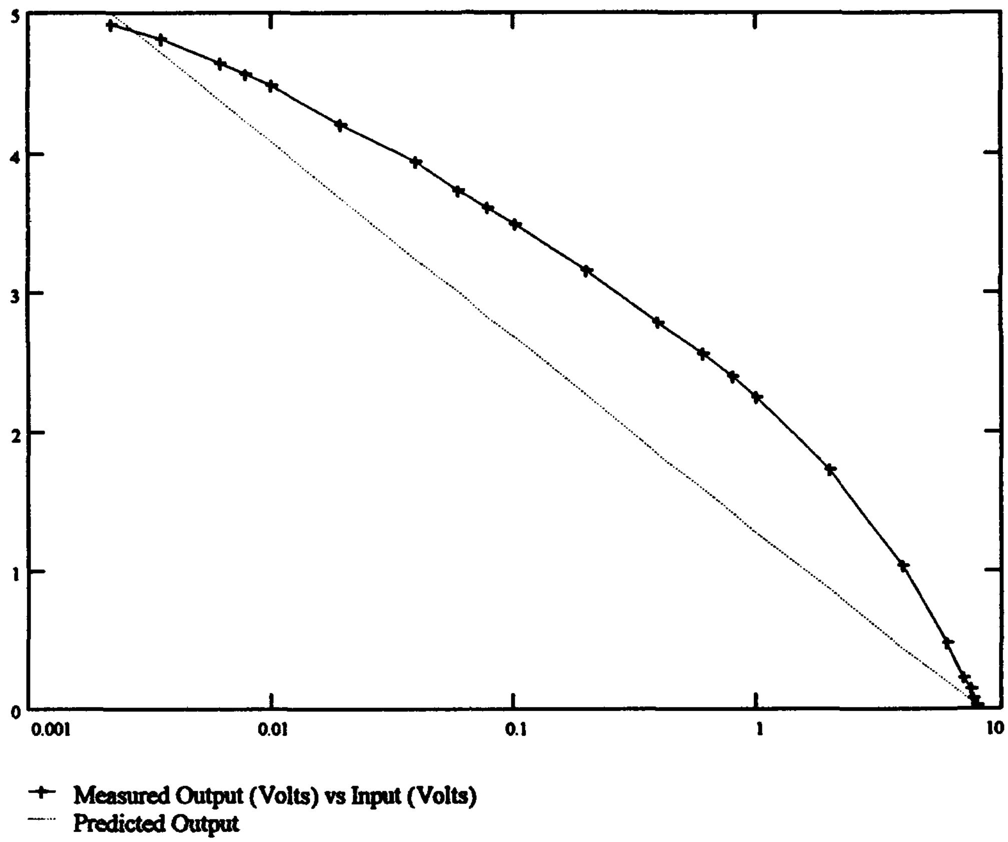


Figure 5-10

From figure 10 we see that the logarithmic compressor does not follow an exact logarithmic relation. However, since the output of the logarithmic compressor monotonically decreases throughout the full range of input values, the compressor is sufficiently logarithmic to achieve the dynamic resolution of equation (5-28). Since we achieve the desired dynamic resolution, the performance of the logarithmic compressor is acceptable.

5.6 Conclusions

In this chapter a receiver to implement the maximum likelihood estimation algorithms developed in chapter II and III was designed and tested. Appendix 1

contains the schematic diagrams of the receiver built for testing. From the tests conducted, we see that the test receiver meets the design criteria developed in section 5.2. To complete the receiver, an Apple Macintosh personal computer is used for the final signal processing and for data display. Appendix 3 contains a listing of the source code used by the Macintosh for signal processing and for displaying the collected data.

Chapter VI

Conclusion

6.1 Observations and Summary

In this thesis, a method to passively localize stationary and linearly moving acoustic beacons was developed. Because of the nonlinearities of the structural model $g(\mathbf{w}_\mu, \theta)$, and the errors in the measurements x_μ, y_μ, and $t_{s\mu}$, the localization problem was formulated as a constrained maximum likelihood estimation problem. To solve for the maximum likelihood estimate, a vector of Lagrange multipliers was introduced and a Lagrangian function Λ formed. The maximum likelihood estimate was then found at the stationary point of the Lagrangian function. Figures 2-4 and 3-1 summarize the algorithms used to solve for the maximum likelihood estimate of the acoustic beacon's position. Both algorithms require that the user specify the maximum range and depth of the search area, and an initial guess of the acoustic beacon's location. For the case of the moving beacon, the user must also specify the maximum allowed velocity of the acoustic beacon.

Monte Carlo simulations were conducted for several different positions and velocities of the acoustic beacon. It was assumed that the receiving ship followed a hexagonal search path with legs of 1000 meters. In practice, the actual search path is not critical as long as it is not linear. In chapter IV results from six scenarios were presented. Overall, the algorithms developed in chapters II and III performed well for these scenarios. For a stationary acoustic beacon, good estimates of the acoustic beacon's location were obtain using GPS errors of 3 meters and 25 meters rms. For

the moving acoustic beacon, a good estimate of the acoustic beacon's location was obtained when using GPS errors of 3 meters rms. However, the accuracy of the algorithm used to estimate the moving acoustic beacon's location degrades for GPS errors of 25 meters rms unless a large number of experiment vectors are used, or a large effective sensor spacing is used.

From the results of the Monte Carlo simulations, it was shown that, in general, as the number of experiment vectors (w_μ) used in the maximum likelihood estimation problem increased, or the effective sensor spacing increased, the accuracy of the estimated beacon location improved. It was also shown that the accuracy of the estimated beacon location improved if the receiving ship passed through a closest point of approach to the acoustic beacon. In all cases, the accuracy of the estimated beacon position was significantly better when using 3 meter GPS errors. This shows that to obtain a better estimate of the acoustic beacon's location, the position of the receiving ship must be measured accurately.

For all scenarios, the variance of the estimated beacon position and velocity decreased as the number of experiment vectors used increased, or the effective sensor spacing increased. It was found that the variance of the moving beacon's estimated position was much greater then the variance of the stationary beacon's estimated position. This is due the manner in which the acoustic beacon's initial position and velocity are combined in the modified structural model. In the modified structural model given by equation 3-10, the individual velocity components are added to the initial beacon coordinates x_o, y_o, and z_o to find the location of the acoustic beacon

when it transmits a signal. As a result, the algorithm estimates the value of the sum of the initial position coordinates and the velocities better then the individual components. Consequently, the algorithm can not discern between small perturbations in the estimated position coordinates and the estimated velocities.

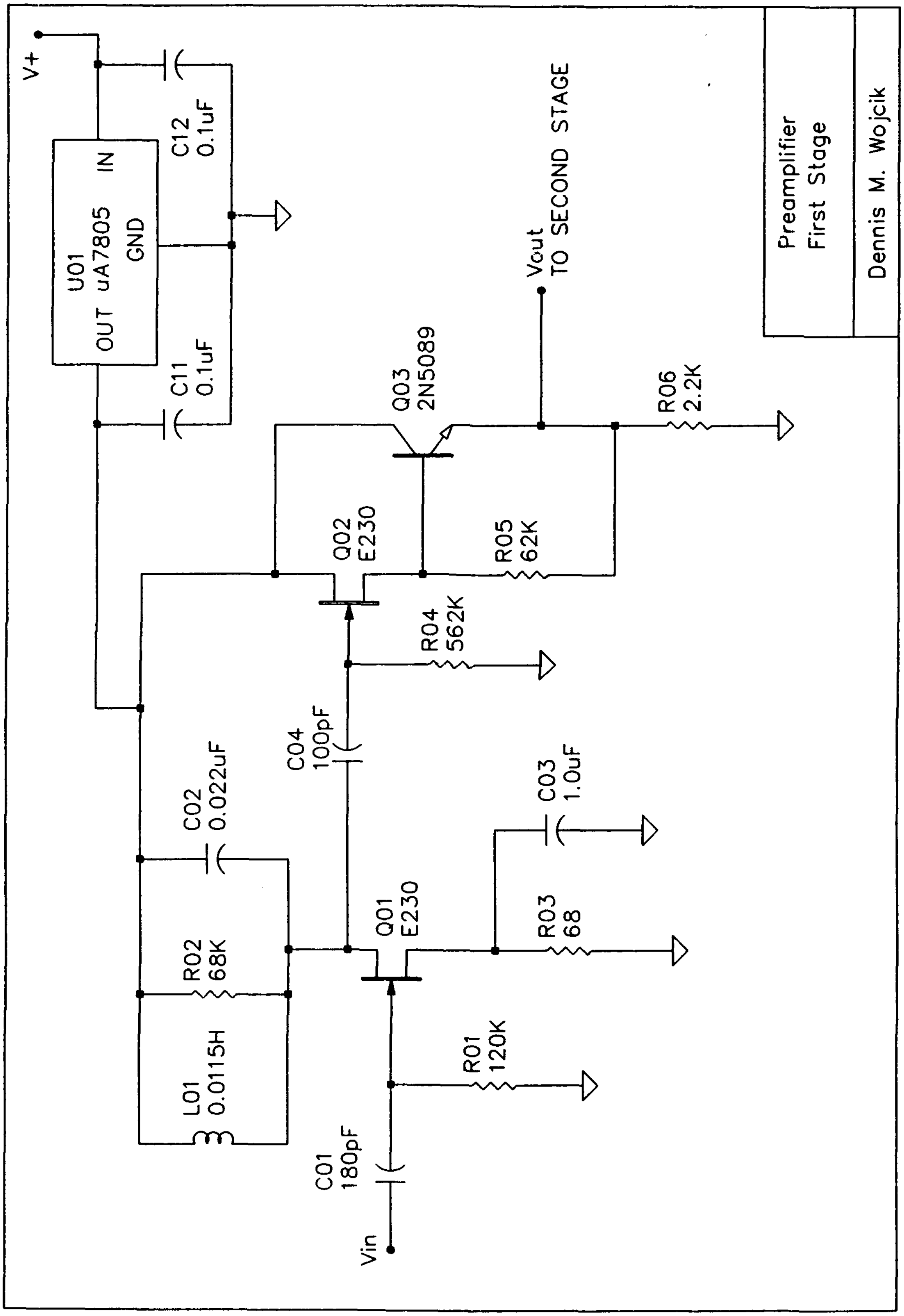
From these results, to obtain the best possible estimate of the acoustic beacon's position a large number of experiment vectors with a large effective sensor spacing must be used. However, the time required to collect the measurement vectors increases with both the number of experiment vectors and the effective sensor spacing. If the time it takes to obtain an estimate of the acoustic beacon's position is an issue, fewer experiment vectors and smaller effective sensor spacings can be used. For the hexagonal search path shown in figure 4-1, and a pulse repetition interval of 2 seconds, a minimum of 20 experiment vectors with an effective sensor spacing of 35 pulse repetition intervals are required to obtain a decent estimate of the acoustic beacon's position for a stationary beacon. For the moving beacon, assuming the same search path, a minimum of 30 experiment vectors with an effective sensor spacing of 45 pulse repetition intervals are required.

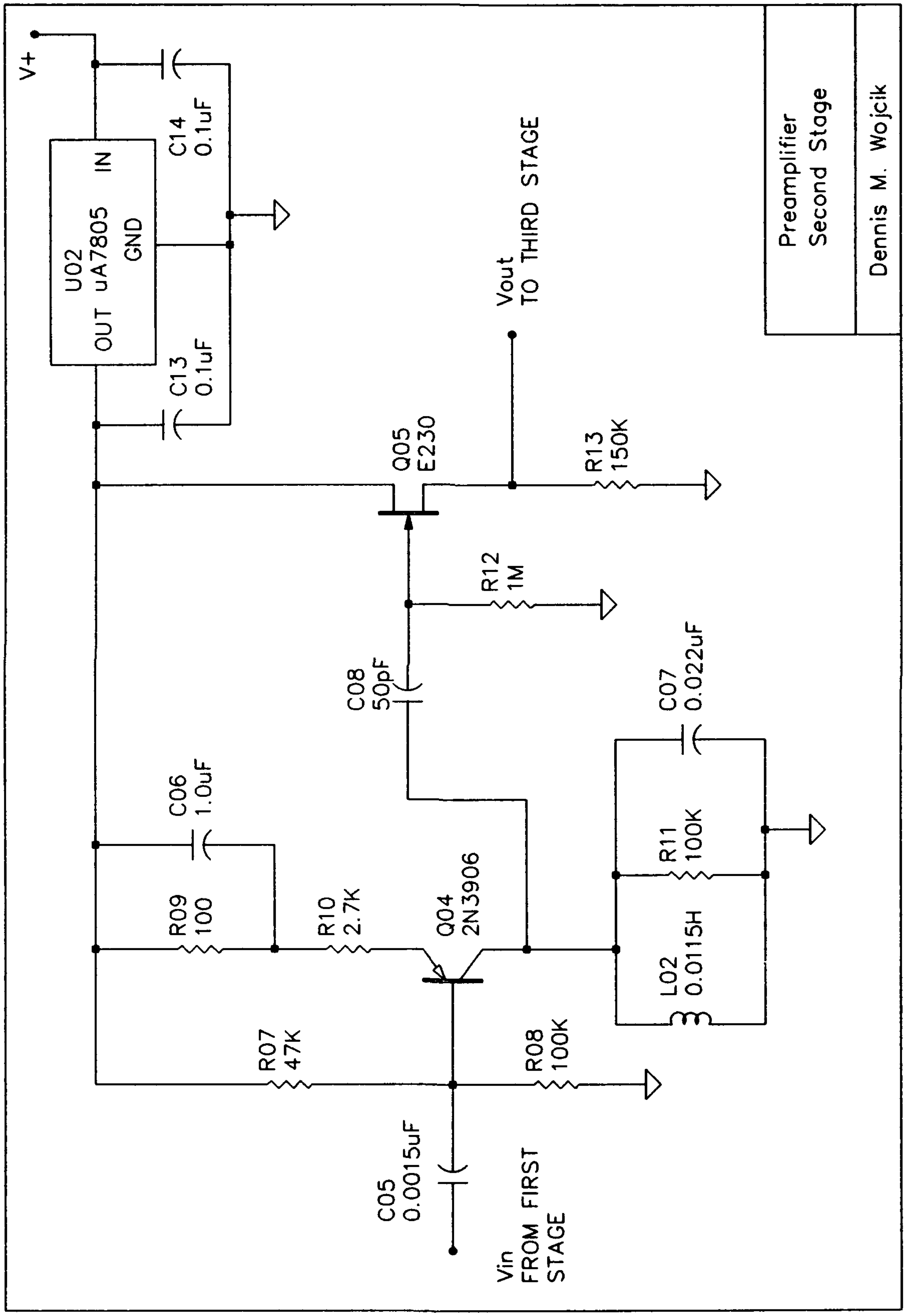
In chapter V design specifications for a receiver to implement the algorithms given in chapters II and III were developed. The synchronous time of receipt was measured by determining the time of arrival of the acoustic beacon's envelope. To determine the arrival time of the acoustic beacon's envelop, the beacon's carrier frequency was first removed using a precision rectifier with a low pass filter to form an envelope detector. The output of the envelope detector was then logarithmically

compressed to increase the dynamic resolution of the receiver. Next, the output of the logarithmic compressor was digitally sampled using a 68HC11 microcontroller. Following A/D conversion, the sampled envelope was passed through a digital matched filtered. For a rectangular windowing function, this produces a unique maximum output which was used to determine the time of arrival. Following the design specifications, a receiver was built from commonly available components. Experiments conducted on receiver showed that the test receiver met the design specifications of section 5.2.

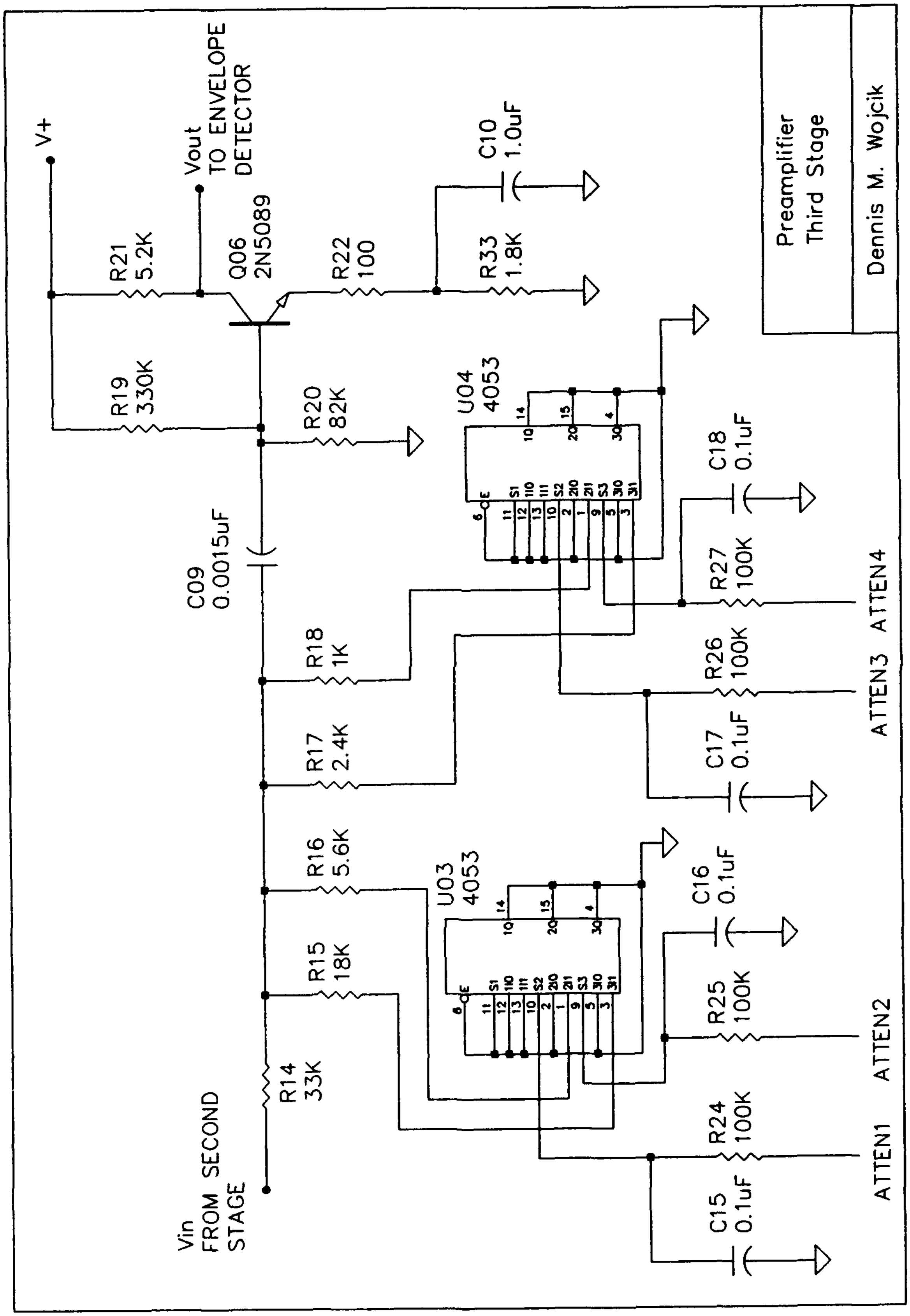
Appendix 1

This appendix contains the schematic diagrams for the receiver designed in chapter V.

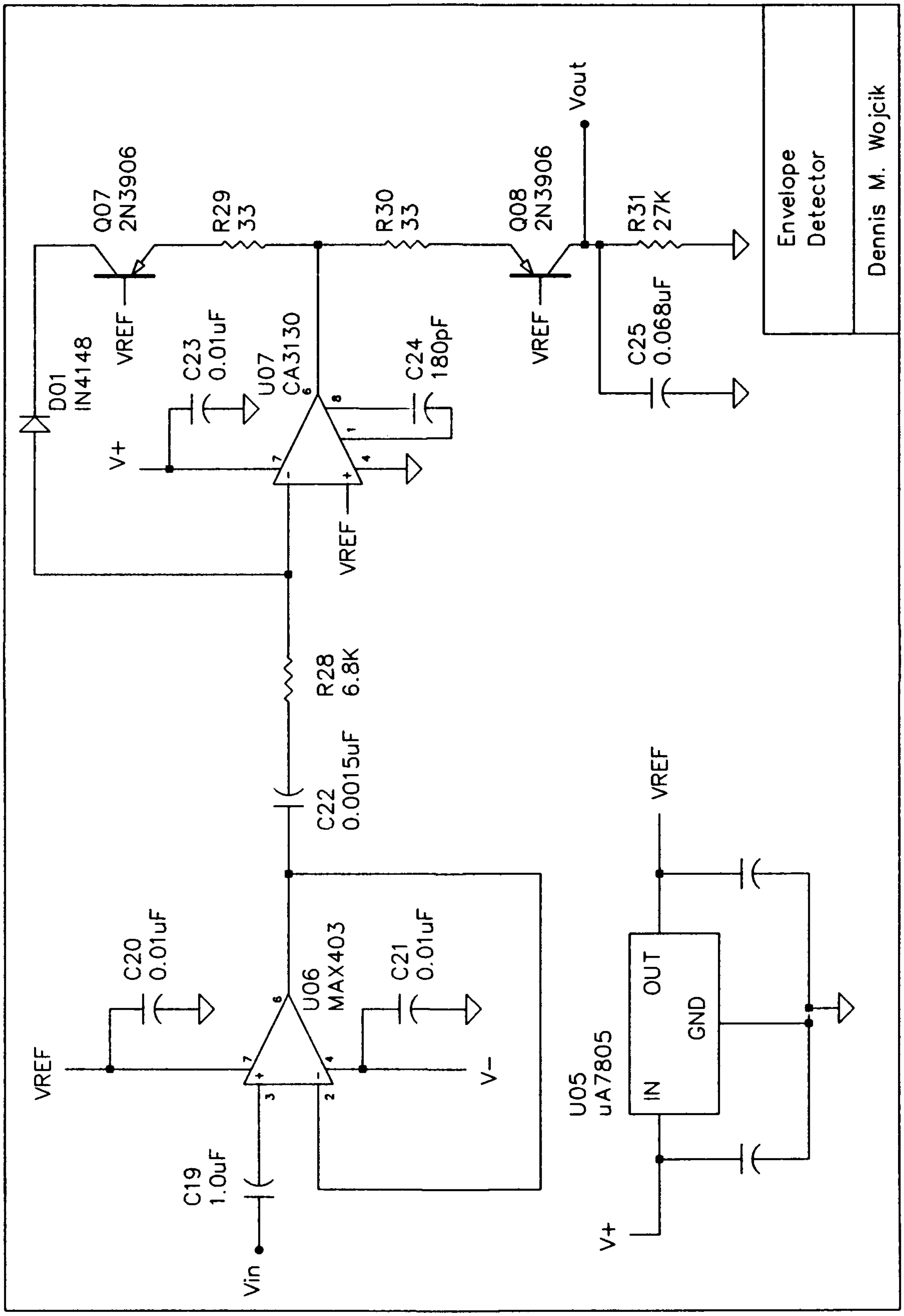

Preamplifier
First Stage
Dennis M. Wojcik
V+
U01
uA7805
OUT
IN
GND
C12
0.1uF
C11
0.1uF
Vout
TO SECOND STAGE
Q03
2N5089
R06
2.2K
Q02
E230
R05
62K
R04
562K
C04
100pF
C03
1.0uF
C02
0.022uF
R02
68K
Q01
E230
R03
68
L01
0.0115H
R01
120K
C01
180pF
Vin

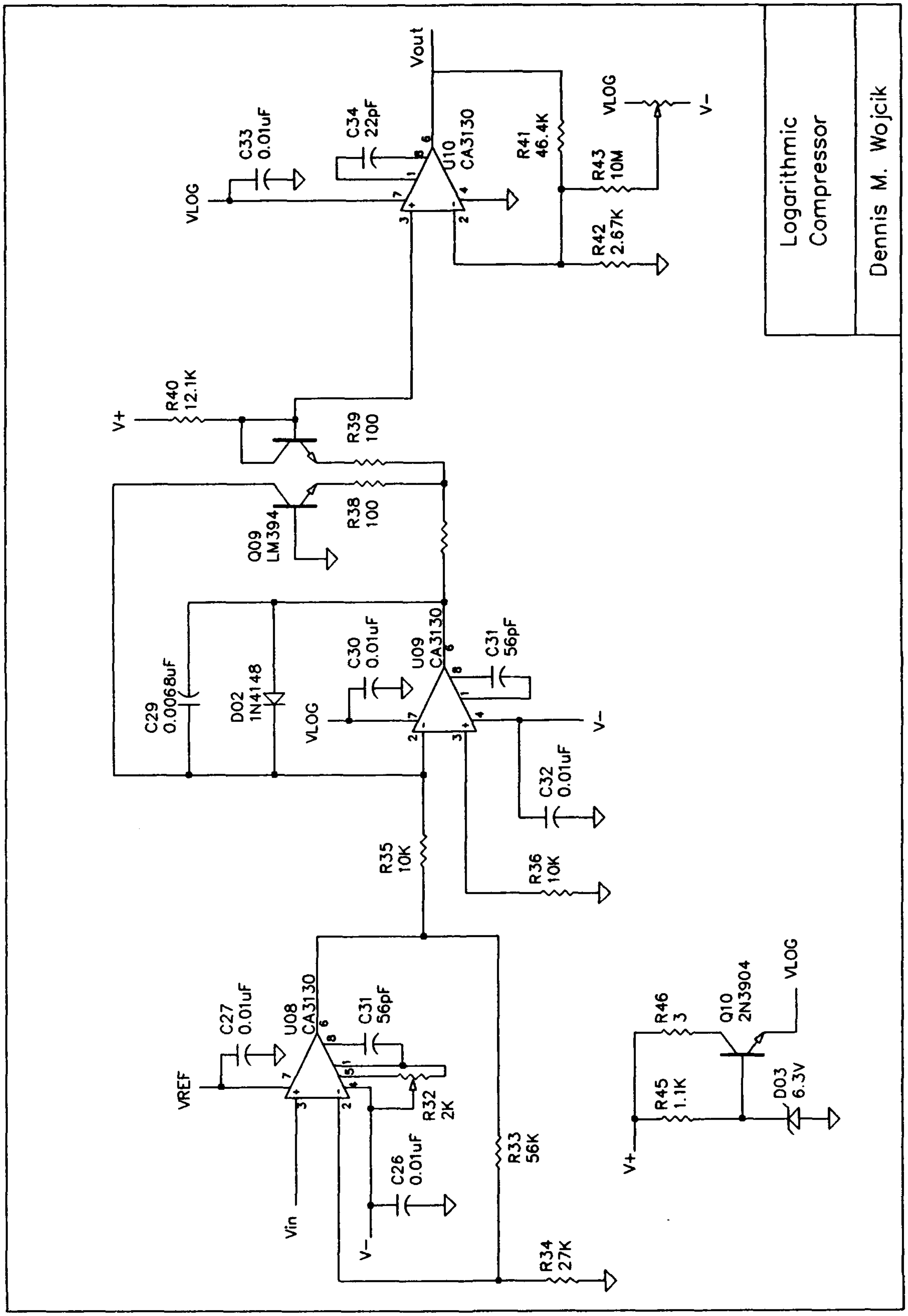


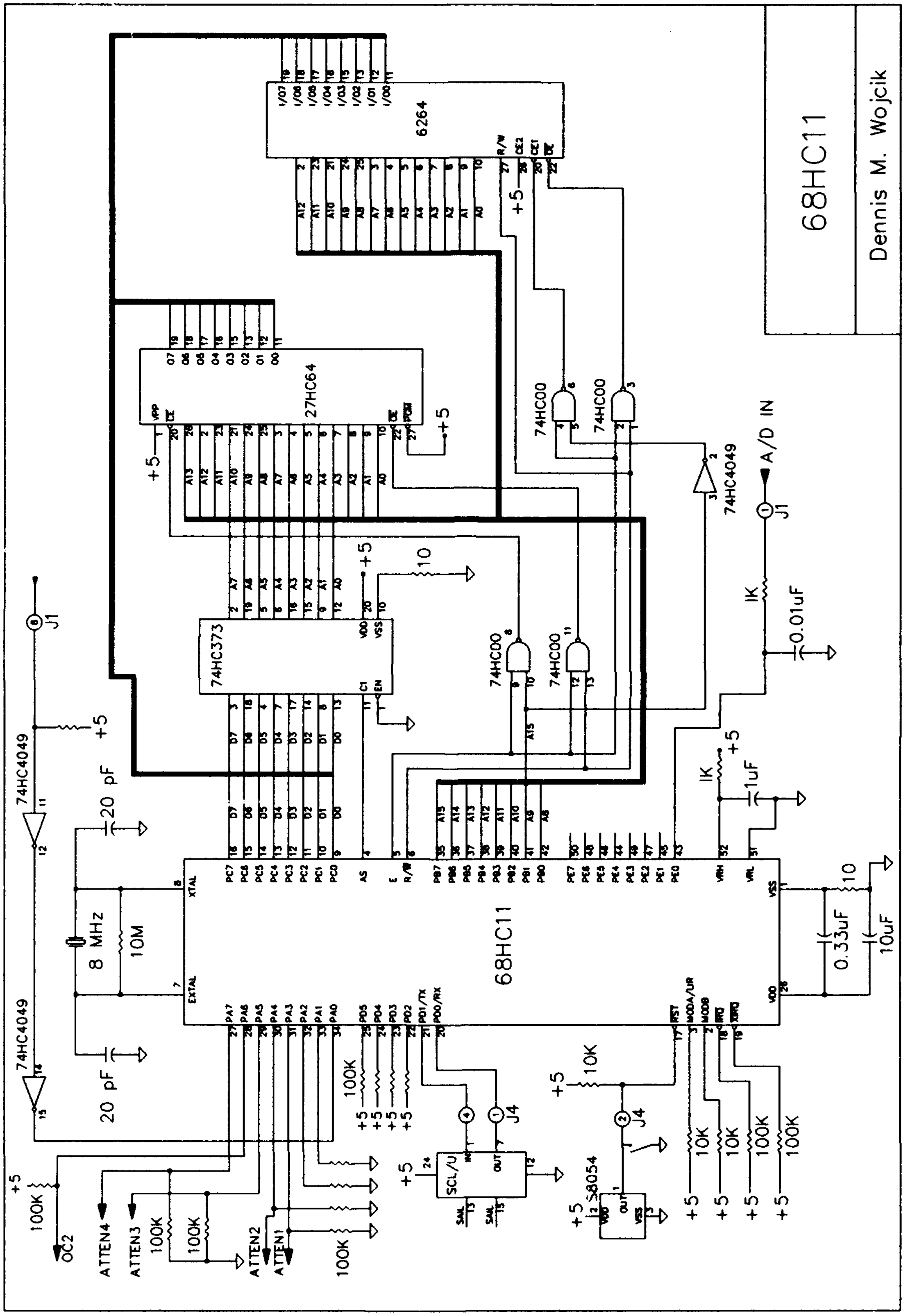
Preamplifier
Second Stage
Dennis M. Wojcik
V+
U02
uA7805
OUT
GND
IN
C13
0.1uF
C14
0.1uF
Vout
TO THIRD STAGE
Q05
E230
R13
150K
R12
1M
C08
50pF
C07
0.022uF
C06
1.0uF
R11
100K
R09
100
R10
2.7K
Q04
2N3906
L02
0.0115H
R07
47K
R08
100K
C05
0.0015uF
Vin
FROM FIRST
STAGE

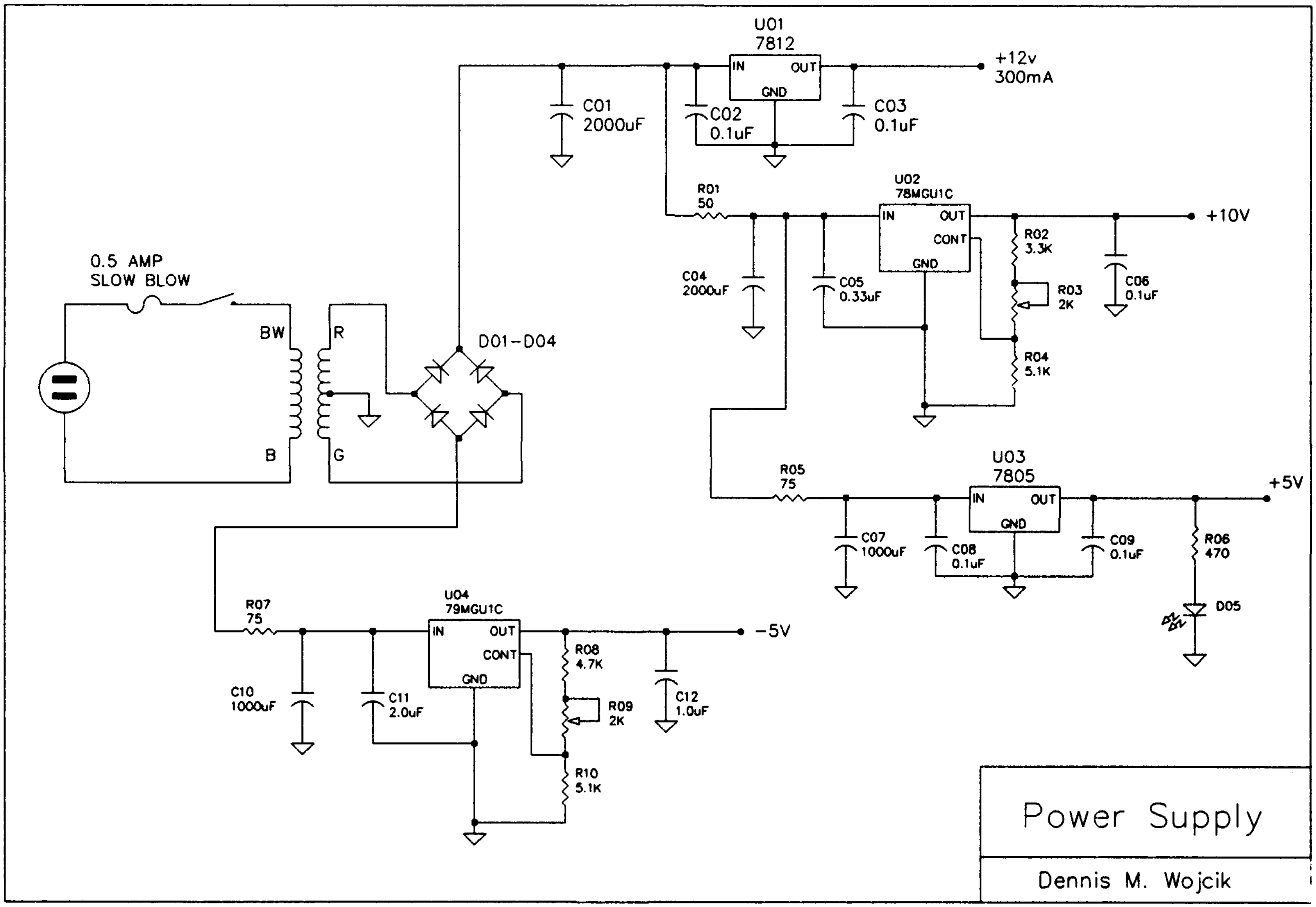


Preamplifier
Third Stage
Dennis M. Wojcik
V+
Vout
TO ENVELOPE
DETECTOR
R21
5.2K
Q06
2N5089
R22
100
R33
1.8K
C10
1.0uF
R19
330K
R20
82K
U04
4053
C18
0.1uF
C09
0.0015uF
R18
1K
R17
2.4K
R27
100K
R26
100K
C17
0.1uF
ATTEN4
ATTEN3
R16
5.6K
U03
4053
C16
0.1uF
R15
18K
R25
100K
ATTEN2
R14
33K
R24
100K
ATTEN1
C15
0.1uF
Vin
FROM SECOND
STAGE


Envelope
Detector
Dennis M. Wojcik
Vout
Q07
2N3906
R29
33
R30
33
Q08
2N3906
R31
27K
VREF
VREF
D01
IN4148
C23
0.01uF
U07
CA3130
C24
180pF
C25
0.068uF
V+
VREF
R28
6.8K
C22
0.0015uF
VREF
C20
0.01uF
U06
MAX403
C21
0.01uF
VREF
V−
OUT
GND
IN
U05
uA7805
C19
1.0uF
Vin
V+


Logarithmic
Compressor
Dennis M. Wojcik
Vout
VLOG
C33
0.01uF
C34
22pF
U10
CA3130
R41
46.4K
R43
10M
R42
2.67K
V−
V+
R40
12.1K
R39
100
R38
100
Q09
LM394
C29
0.0068uF
D02
1N4148
C30
0.01uF
U09
CA3130
C31
56pF
C32
0.01uF
R35
10K
R36
10K
VREF
C27
0.01uF
U08
CA3130
C31
56pF
R32
2K
C26
0.01uF
R33
56K
Vin
R34
27K
R46
3
Q10
2N3904
R45
1.1K
D03
6.3V


68HC11
Dennis M. Wojcik
6264
27HC64
74HC373
74HC00
74HC4049
SCL/U
S8054
A/D IN
ATTEN1
ATTEN2
ATTEN3
ATTEN4
OC2
8 MHz
10M
20 pF
0.33uF
10uF
0.01uF
1uF


0.5 AMP
SLOW BLOW
BW
B
R
G
D01-D04
C01
2000uF
U01
7812
IN
OUT
GND
C02
0.1uF
C03
0.1uF
+12v
300mA
R01
50
C04
2000uF
C05
0.33uF
U02
78MGU1C
IN
OUT
CONT
GND
R02
3.3K
R03
2K
R04
5.1K
C06
0.1uF
+10V
R05
75
C07
1000uF
C08
0.1uF
U03
7805
IN
OUT
GND
C09
0.1uF
R06
470
D05
+5V
R07
75
C10
1000uF
C11
2.0uF
U04
79MGU1C
IN
OUT
CONT
GND
R08
4.7K
R09
2K
R10
5.1K
C12
1.0uF
-5V
Power Supply
Dennis M. Wojcik

Appendix 2

This appendix contains the source code used by the 68HC11. The code is based on a full function SAIL driver written by Dr. Albert Bradley and operates under serial communication interrupt control. The sampling rate of the A/D converter is set by using a Timer Output Compare function. After the microcontroller finishes an initialization period, it sits in a wait state until a serial communications interrupt or a timer output compare interrupt occurs. Once the interrupt is received, the microcontroller executes the requested task, and then returns to the wait state.

```
; #PTR Driver
;
;THE MINIMUM POWER DRAIN IS-
;       2.5 MA IF IN WAIT LOOP, 4.9152 MHZ
;       6-7 MA IF WORKING, NO LOADS
;*****************************************************************************
; Mon, Sep 17, 1990, 19:21 MINOR CLEANUPS
; IT CONTAINS-
;       -?M AND !M ARE OPTIONAL DEPENDING ON THE 'BIG' FLAG
;       -FULL BAUD SWITCH CAPABILITY
;       -FULL USE OF EVEN PARITY
;       -FULL TTY HANDLES
;*****************************************************************************
        PROC  "68HC11"
; THIS IS THE CONTROL FOR THE CONDITIONAL ASSEMBLY OF THE !M AND ?M
SECTIONS
;         =1 WILL INCLUDE THEM
;         =0 WILL EXCLUDE THEM (SAVES $118 BYTES OF CODE)
LOAD:           EQU     1
VIEW:           EQU     1
;
;*****************************************************************************
;
; DEFAULT PARAMETERS
DBAUD:          EQU     $30             ;START UP AT 9600 BAUD WITH 8.0 MHZ XTAL
SAMPLE:         EQU     $07D1           ;1.0mSEC SAMPLING RATE (ADJUSTED FOR ERRORS)
RADBUFF:        EQU     $3000           ;LOCATION OF RUNNING A/D BUFFER
ADBFULL:        EQU     $300B           ;A/D BUFFER FULL AT 11 VALUES (0-10)
MAXBUF1:        EQU     $3100           ;LOCATION OF BUFFER 1
MAXBUF2:        EQU     $3200           ;LOCATION OF BUFFER 2
```

```
ATTNINC:       EQU   $E8       ;INCREASE ATTN AT 4.75 VOLTS
ATTNDEC:       EQU   $40       ;INCREASE THE PREAMP GAIN AT THIS LEVEL
MAX1FL:        EQU   $310F     ;CHECK VALUE TO INITIALIZE MAX BUFFER
MAX2FL:        EQU   $320F     ;CHECK VALUE TO INITIALIZE MAX BUFFER
;
; ATTENUATOR LEVELS
DB00:          EQU   $01       ; 0 DB ATTEN
DB03:          EQU   $09       ; 3 DB ATTEN
DB09:          EQU   $11       ; 9 DB ATTEN
DB12:          EQU   $19       ;12 DB ATTEN
DB18:          EQU   $21       ;18 DB ATTEN
DB24:          EQU   $81       ;24 DB ATTEN
DB30:          EQU   $B9       ;30 DB ATTEN
;
DMAXNUM:       EQU   $02       ;DEFAULT NUMBER OF MAX POINTS TAKEN
DPERIOD:       EQU   $07D0     ;DEFAULT TO 2 SECOND PERIOD
DWIND:         EQU   $0014     ;DEFAULT INDEX WINDOW
;
;****************************************************************************
;
RP:            EQU   $0000     ;REGISTER PAGE ON BASE PAGE
PORTA:         EQU   RP+$00    ;|OC1.OC2.OC3.OC4!OC5.IC1.IC2.IC3|
PORTC:         EQU   RP+$03    ;
PORTB:         EQU   RP+$04    ;
DDRC:          EQU   RP+$07    ;
PORTD:         EQU   RP+$08    ;|X.X.SS.SCK!MOSI.MISO.TXD.RXD|
DDRD:          EQU   RP+$09    ;
PORTE:         EQU   RP+$0A    ;|V7.V6.V5.V4!X.X.X.BRK|
;                              ;BAT.BAUD.
CFORC:         EQU   RP+$0B    ;TIMER COMPARE FORCE REGISTER
OC1M:          EQU   RP+$0C    ;OUT COMP 1 MASK REG
OC1D:          EQU   RP+$0D    ;OUT COMP 1 DATA REG
TCNT:          EQU   RP+$0E    ;TIMER REGISTER (HIGH BYTE)
DTIC1:         EQU   RP+$10    ;TIMER INPUT CAPTURE 1 (HI BYTE)
DTIC2:         EQU   RP+$12    ;TIMER INPUT CAPTURE 2 (HI BYTE)
DTIC3:         EQU   RP+$14    ;TIMER INPUT CAPTURE 3 (HI BYTE)
DTOC1:         EQU   RP+$16    ;TIMER OUTPUT COMPARE 1 (HI BYTE)
DTOC2:         EQU   RP+$18    ;TIMER OUTPUT COMPARE 2 (HI BYTE)
DTOC3:         EQU   RP+$1A    ;TIMER OUTPUT COMPARE 3 (HI BYTE)
DTOC4:         EQU   RP+$1C    ;TIMER OUTPUT COMPARE 4 (HI BYTE)
DTOC5:         EQU   RP+$1E    ;TIMER OUTPUT COMPARE 5 (HI BYTE)
TCTL1:         EQU   RP+$20    ;TIMER CONTROL REG 1
TCTL2:         EQU   RP+$21    ;TIMER CONTROL REG 2
TMSK1:         EQU   RP+$22    ;TIMER INTERRUPT MASK REG 1
TFLG1:         EQU   RP+$23    ;TIMER INTERRUPT FLAG REG 1
TMSK2:         EQU   RP+$24    ;TIMER INTERRUPT MASK REG 2
TFLG2:         EQU   RP+$25    ;TIMER INTERRUPT FLAG REG 2
PACTL:         EQU   RP+$26    ;PULSE ACCUMULATOR CONTROL REG
PACNT:         EQU   RP+$27    ;PULSE ACCUMULATOR COUNT REG
SPCR:          EQU   RP+$28    ;SPI CONTROL REGISTER
SPSR:          EQU   RP+$29    ;SPI STATUS REGISTER
```

```
SPDR:       EQU  RP+$2A    ;SPI DATA REGISTER
BAUD:       EQU  RP+$2B
SCCR1:      EQU  RP+$2C
SCCR2:      EQU  RP+$2D
SCSR:       EQU  RP+$2E
SCDR:       EQU  RP+$2F
ADCTL:      EQU  RP+$30
ADR1:       EQU  RP+$31
ADR2:       EQU  RP+$32    ;
ADR3:       EQU  RP+$33    ;
ADR4:       EQU  RP+$34    ;

OPTION:     EQU  RP+$39
HPRIO:      EQU  RP+$3C
INIT:       EQU  RP+$3D    ;[RAM3.2.1.0!REG3.2.1.0]
;
;*********************************************************************
; SAIL inteface RAM workspace
RJUMP:      EQU  $40       ;JUMP INST GOES HERE..
RVECT:      EQU  $41       ;      HI
;                $42       ;      LO
XJUMP:      EQU  $43       ;XMIT JUMP INST
XVECT:      EQU  $44       ;      HI
;                $45       ;      LO
JUMP2:      EQU  $46       ;2ND  JUMP INST
VECT2:      EQU  $47       ;      HI
;                $48       ;      LO
SCRTCH1:    EQU  $49       ;SCRATCH LOC 1
SCRTCH2:    EQU  $4A       ;SCRATCH LOC 2
SCRTCH3:    EQU  $4B       ;SCRATCH LOC 3
SCRTCH4:    EQU  $4C       ;SCRATCH LOC 4
SCRTCH5:    EQU  $4D       ;SCRATCH LOC 5
;
XTTY:       EQU  $4E       ;TTY POINTER HI
;                $4F              LOW
;
;*****************************************************************
; PTR Data inteface RAM workspace
PERINDX:    EQU  $50       ;PERIOD TIMER INDEX HI
;                $51                  LOW
PERIOD:     EQU  $52       ;LENGTH OF PERIOD HI
;                $53                LOW
MAXBUFF:    EQU  $54       ;MAXIMUMS BUFFER LOCATION HI
;                $55                         LOW
MAXFULL:    EQU  $56       ;MAX BUFFER FULL LOCATION HI
;                $57                         LOW
ADBFPTR:    EQU  $58       ;RUNNING A/D BUFFER LOCATION HI
;                $59                            LOW
TXBUFF:     EQU  $5A       ;MAXIMUMS BUFFER TO TX HI
;                $5B                   LOW
TXBFULL:    EQU  $5C       ;TX BUFFER FULL LOCATION HI
```

```
;               $5D                     LOW
TX1FLG:    EQU  $5E          ;FIRST MAXIMUMS SENT FLAG
TX2FLG:    EQU  $5F          ;SECOND MAXIMUMS SENT FLAG
;
MAXFLG:    EQU  $60          ;WHICH MAXIMUM BUFFER TO SAVE FLAG
AGCFLG:    EQU  $61          ;MANUAL GAIN CONTROL FLAG
BIG:       EQU  $62          ;LARGEST A/D VALUE PER PERIOD LOCATION
MAXNUM:    EQU  $63          ;NUMBER OF MAXIMUMS LOCATION
TXMHI:     EQU  $64          ;TRANSMIT OUT BUFFERS
TXMLO:     EQU  $65
TXIHI:     EQU  $66
TXILO:     EQU  $67
TXINT:     EQU  $68          ;INTERM STORAGE FOR  TX POINTER
TEMPIND:   EQU  $70          ;INTERM STORAGE FOR MAXIMUM INDEX
TEMPMAX:   EQU  $72          ;INTERM STORAGE FOR MAXIMUM
WINDOW:    EQU  $74          ;MAXIMUM CHECK WINDOW SIZE
WINCHK:    EQU  $76          ;USED FOR TESTING IF WITHIN WINDOW
INDXCHK:   EQU  $78          ;
;
;**********************************************************************
      ORG   $E800  ;ASSUME 27C64 EPROM
; SAIL Address stored here..
ADDR:      FCB  "#PTR"       ;SAIL ADDRESS
           FCB  $00          ;TERMINATOR
;
;**********************************************************************
START:     SEI
           CLR  $103D        ;MOVE REGISTERS TO BASE PAGE (00-> INIT)
           LDS  #$00FF       ;SET STACK POINTER
           LDAA  #DBAUD      ;SET DEFAULT BAUD
           STAA  BAUD
           CLR  SCCR1        ;8 DATA BITS, WAKE ON IDLE?
           LDAA  #$2C        ;ONLY REC INTERRUPTS YET
           STAA SCCR2
;**********************************************************************
; PTR TIMER INTERUPT INITIALIZATION
           LDAA #$40         ;TOGGLE OC2 ON EACH COMPARE
           STAA TCTL1
           CLR  TMSK2        ;SET PRESCALE TO /1
           LDAA #$90         ;TURN ON THE A/D CONVERTER
           STAA OPTION
           LDAA #$20         ;USE PEO FOR A/D INPUT, SINGLE CHANNEL,
           STAA ADCTL        ;CONTINUOUS SAMPLING
           LDAA #$88         ;SET PORT A PINS 7 AS OUTPUT
           STAA PACTL
           CLR  MAXFLG       ;INITIALIZE TO FIRST MAXIMUMS BUFFER
           CLR  TX1FLG       ;INITIALIZE TO NOT TX'D
           CLR  TX2FLG       ;INITIALIZE TO NOT TX'D
           CLR  AGCFLG       ;INITIALIZE TO AGC ON
           CLR  BIG          ;INITIALIZE THE LARGEST VALUE TO ZERO.
           CLR  TXMHI        ;CLEAR THE OUTGOING BUFFERS
```

```
                CLR     TXMLO
                CLR     TXIHI
                CLR     TXILO
                LDAA    #DB00           ;SET 0 DB OF ATTENUATION
                STAA    PORTA
                LDD     #RADBUFF        ;SET A/D DATA BUFFER LOCATION IN BASE PAGE
                STD     ADBFPTR
                LDX     #MAXBUF1        ;SET FIRST MAX BUFFER LOCATION IN BASE PAGE
                STX     MAXBUFF
                STX     TXBUFF
                LDAB    #DMAXNUM
                ADDB    #DMAXNUM
                STAB    MAXNUM          ;SET THE NUM MAX CHECK VALUE
                ABX
                STX     MAXFULL         ;SET BUFFER FULL TO 2xDMAXNUM
                STX     TXBFULL
                LDD     #DPERIOD        ;SET THE PERIOD SIZE
                STD     PERIOD
                LDX     #$0000
                STX     PERINDX         ;INITIALIZE PERIOD TIMER TO ZERO
                LDAA    #$00
                LDX     ADBFPTR
CLRAD:          STAA    $00,X           ;CLEAR THE RUNNING A/D BUFFER
                INX
                CPX     #ADBFULL
                BLO     CLRAD
                LDX     #MAXBUF1        ;CLEAR THE FIRST MAXIMUM BUFFER
CLMX1:          STAA    $00,X
                INX
                CPX     #MAX1FL
                BLS     CLMX1
                LDX     #MAXBUF2        ;CLEAR THE SECOND MAXIMUM BUFFER
CLMX2:          STAA    $00,X
                INX
                CPX     #MAX2FL
                BLS     CLMX2
                LDD     #DWIND          ;SET DEFAULT INDEX WINDOW
                STD     WINDOW
;
;*********************************************************************
;
                CLR     SCRTCH1         ;USE AS SAIL MODE FLAG & ADDR POINTER
                LDAA    #$7E            ;MUST WRITE JMP INSTS.
                STAA    RJUMP           ;INTO SAIL WORK AREA
                STAA    XJUMP
                STAA    JUMP2
                LDD     #SETUNAD        ;SET VECTORS TO INITIAL VALUES
                STD     RVECT
                STD     XVECT
                CLI                     ;AND FINALLY ALLOW INTERRUPTS
;
;*********************************************************************
;
```

```
;***************************************************************
; MAIN PROGRAM BODY
LOOP:       WAI                 ;WAIT TILL INTERRUPTED..
            BRA   LOOP
;***************************************************************
;***************************************************************
;
;  TIMER INTERUPT HANDLER
;       Fri, Jul 10, 1992, 13:02
;       Wed, Jul 22, 1992, 8:35
;       Fri, Apr 23, 1993, 9:12
;       Thu, May 20, 1993, 1:52 PM
;
OC2:        LDD   DTOC2         ;GET LAST VALUE IN TOC2
            ADDD  #SAMPLE       ;AND ADD ONE PERIOD TO IT
            STD   DTOC2         ;STORE BACK IN TOC2
;
            LDAA  #$FF          ;NORMALIZE THE A/D VALUE
            SUBA  ADR1
            CMPA  BIG           ;IS A/D VALUE GREATER THEN BIG?
            BLO   TI1           ;IF IT IS STORE IN BIG ELSE MOVE ON
            STAA  BIG
;
TI1:        LDX   ADBFPTR       ;GET THE CURRENT A/D POSITION
            STAA  $00,X         ;STORE THE VALUE
            INX                 ;UPDATE THE POINTER
            CPX   #ADBFULL      ;CHECK IF AT LAST A/D STORAGE LOCATION
            BLO   TI2           ;IF NOT INC ADBPTR AND MOVE ON
            LDX   #RADBUFF      ;IF YES REINITIALIZE ADBPTR AND MOVE ON
TI2:        STX   ADBFPTR       ;SAVE THE A/D STORAGE POINTER
;
            LDX   #RADBUFF      ;GET THE A/D BUFFER LOCATION
            LDY   #$0000        ;INITIALIZE THE SUMATION VALUE
SUMMING:    LDAB  $00,X         ;GET THE A/D VALUE TO ADD
            ABY                 ;SUM=SUM+A/D
            INX                 ;CONTINUE UNTIL ALL VALUES IN THE A/D
            CPX   #ADBFULL      ;BUFFER ARE SUMMED
            BLO   SUMMING
;
            LDX   MAXBUFF       ;GET THE BUFFER TO WRITE TO
            LDD   PERINDX       ;SETUP FOR TESTING MAX VALUES
            STD   TEMPIND
            CPD   WINDOW        ;SEE IF THE MAX IS CLOSE TO THE START OF
;                                   THE PERIOD
            BHI   TI1_1_1       ;IF NOT MOVE ON, OTHERWISE
            CPY   $00,X         ;CHECK FOR A NEW MAXIMUM
            BLO   TI1_2         ;IF NOT GOTO CHECK FOR SECONDARY MAX'S
            STY   $00,X         ;IF YES JUST REPLACE CURRENT MAX
            LDD   TEMPIND
            STD   $10,X
            BRA   TIENDJ
```

```
;
TI1_1_1:    ADDD  WINDOW     ;SEE IF THE MAX IS CLOSE TO THE END OF PERIOD
            CPD   PERIOD
            BLO   TI1_1_2    ;IF NOT MOVE ON, OTHERWISE
            CPY   $00,X      ;CHECK FOR A NEW MAXIMUM
            BLO   TI1_2      ;IF NOT GOTO CHECK FOR SECONDARY MAX'S
            STY   $00,X      ;IF YES JUST REPLACE CURRENT MAX
            LDD   TEMPIND
            STD   $10,X
            BRA   TIENDJ
;
TI1_1_2:    LDD   PERINDX
            SUBD  WINDOW
            STD   WINCHK
            CPY   $00,X      ;CHECK FOR A NEW MAXIMUM
            BLO   TI1_2      ;IF NOT CHECK FOR SECONDARY MAX'S
            CPD   $10,X      ;CHECK IF WITHIN WINDOW
            BHI   TI1_1      ;IF NO REPLACE ALL MAX'S
            STY   $00,X      ;IF YES JUST REPLACE CURRENT MAX
            LDD   TEMPIND
            STD   $10,X
TIENDJ:     BRA   TIEND
;
TI1_1:      LDD   $00,X      ;SHIFT ALL OLD MAX'S ONE LOCATION AND
            STY   $00,X      ;SAVE THE NEW MAX VALUE
            STD   TEMPMAX
            LDY   $10,X      ;SHIFT THE INDICIES
            LDD   TEMPIND
            STD   $10,X
            STY   TEMPIND
            LDY   TEMPMAX
            INX              ;INCREMENT TO THE NEXT MAX
            INX
            CPX   MAXFULL    ;SEE IF DONE
            BLO   TI1_1
            BRA   TIEND
;
TI1_2:      LDD   $10,X      ;CHECK FOR SECONDARY MAX'S
            ADDD  WINDOW     ;SET UP INDEX WINDOW
            STD   INDXCHK
            INX              ;MOVE TO THE NEXT MAX
            INX
            CPX   MAXFULL    ;SEE IF DONE
            BHS   TIEND
            CPY   $00,X      ;LOOK FOR NEW SECONDARY
            BLO   TI1_2
            LDD   TEMPIND
            CPD   INDXCHK    ;IF WITHIN THE WINDOW OF THE LARGER MAX
            BLO   TIEND      ;IGNORE THE VALUE
            LDD   PERIOD
            SUBD  PERINDX
```

```
        CPD    WINDOW
        BLO    TIEND          ;IF ON THE SCREEN WRAP IN WINDOW IGNORE
;
        LDD    PERINDX
        SUBD   WINDOW
        CPD    $10,X          ;IF WITHIN THE WINDOW OF THE SECONDARY
        BHI    TI1_3
        STY    $00,X          ;JUST REPLACE THE ONE VALUE
        LDD    PERINDX
        STD    $10,X
        BRA    TIEND
;
TI1_3:  LDD    $00,X          ;ELSE REPLACE AND SHIFT OLD VALUES DOWN
        STY    $00,X
        STD    TEMPMAX
        LDY    $10,X
        LDD    TEMPIND
        STD    $10,X
        STY    TEMPIND
        LDY    TEMPMAX
        INX
        INX
        CPX    MAXFULL
        BLO    TI1_3
;
TIEND:  LDX    PERINDX        ;GET THE PERIOD COUNTER AND INCREMENT
        INX
        CPX    PERIOD         ;IS THE PERIOD COMPLETED?
        BLO    TI11
;
        INC    MAXFLG
        BRCLR MAXFLG,$01,TI7          ;ARE WE IN MAXIMUMS BUFFER 1 OR 2?
        CLR    TX1FLG         ;CLEAR BUFFER 1 TX'D FLAG
        LDY    #MAXBUF2
        BRA    TI8
TI7:    CLR    TX2FLG         ;CLEAR BUFFER 2 TX'D FLAG
        LDY    #MAXBUF1
TI8:    STY    MAXBUFF        ;UPDATE THE BUFFER TO USE
        LDAB   MAXNUM
        ABY
        STY    MAXFULL        ;UPDATE THE FULL BUFFER VALUE
;
        BRSET AGCFLG,$01,TI10         ;IF MANUAL GAIN SET DON'T CHECK AGC
        LDAA   BIG            ;TEST IF GAIN NEEDS TO BE DECREASED
        CMPA   #ATTNINC       ;IF NOT MOVE ON
        BLO    TI9
        JSR    ATTN           ;GO TO ATTN FOR MORE ATTENUATION
        BRA    TI10
TI9:    CMPA   #ATTNDEC       ;TEST IF GAIN NEEDS TO BE INCREASED
        BHI    TI10           ;IF NOT MOVE ON
        JSR    INC            ;GO TO INC FOR LESS ATTENUATION
```

```
TI10:           CLR    BIG             ;RESET BIG FOR NEXT PERIOD
;
                LDX    #$0000          ;CLEAR THE PERIOD COUNTER
TI11:           STX    PERINDX
                LDAA   #$40            ;RESET TIMER FLAGS
                STAA   TFLG1
                RTI
;
;*********************************************************
;
; SCI INTERRUPT HANDLER (SAIL DRIVER)
;       Sat, May 12, 1990, 20:34 COMPACT BY 20 BYTES BY USING RSET:
;       Sun, May 13, 1990, 12:36 COMPACT BY USING CONDITIONAL ASSEMBLY
;       Sun, May 13, 1990, 12:37  AND BY USING TTY FOR PROMPT
;       Mon, Sep 17, 1990, 19:20 ADD IN4 TO TTY
SCI:            BRCLR SCSR,$0E,CKCHAR       ;TEST FOR VALID CHAR
;&              LDAA   SCSR            ;TEST OR.NF.FE
;               ANDA   #$0E
;               BEQ    CKCHAR          ;IF GOOD, GO ON..
                LDAA   SCDR            ;READ BAD CHAR TO CLEAR?
ECHO:           NOP                    ;  RECEIVE AN ECHO CHARACTER (& IGNORE!)
SCIXIT:         RTI                    ; & EXIT
;----------
;
CKCHAR:         BRSET  SCSR,$20,READD
;&              LDAA   SCSR            ;TEST DA
;               ANDA   #$20
;               BNE    READD
                JMP    CHECKX          ;IF NOT, XMIT
;
READD:          LDAA   SCDR            ;READ DATA TO A..
                TAB                    ;AND COPY TO B
;               LDX    #PARITY         ;POINT PARITY TABLE
;               ABX                    ;POINT INTO PARITY TABLE
;               LDAB   0,X             ;GET PARITY WORD
;               BNE    SCIXIT          ;REJECT IF NOT EVEN PARITY
;               ANDA   #$7F            ;ELSE REMOVE PARITY FOR SUBSEQUENT TESTS
                TAB                    ;(ALSO COPY TO B)
                CMPA   #'#'            ;IS IT '#'?
                BEQ    GOTNUM
                JMP    RJUMP           ;ELSE GO THRU VECTOR
;
ATXIT:          JMP    SCIXIT          ;NO FURTHER USE FOR THIS CHAR, EXIT
;
; SAIL ADDRESS RECOGNIZE SECTION
GOTNUM:         LDAB   #01             ; GOT # CODE
                STAB   SCRTCH1         ;SET SAIL FLAG TO 01
                BCLR   SCCR2,$D2       ;TURN OFF XMIT INTERRUPTS
;&              LDAA   SCCR2
;               ANDA   #$2C            ;TURN OFF XMIT INTERRUPTS
;               STAA   SCCR2
                LDD    #READAD         ;POINT READ ADDRESS
```

```
RSET:       STD   RVECT       ;..MAY ENTER HERE FROM MANY OTHER PLACES
            JMP   SCIXIT      ;AND EXIT
;
READAD:     LDAB  SCRTCH1     ;GET SMODE COUNT
            CMPB  #$01        ;IS THIS THE FIRST CHAR?
            BNE   TRYADD
            CMPA  #'0'        ;IF FIRST, IS IT '0'?
            BEQ   TSTBAUD     ;IF SO, MAY BE BAUD SWITCH
;                             ;CONTINUE ON BELOW...
TRYADD:     LDX   #ADDR               ;ELSE POINT X TO ADDR
            ABX               ;ADD TO X
            CMPA  0,X         ;COMPARE NEW CHAR TO A
            BEQ   NEXTAD      ;IF MATCH, LOOK FOR NEXT IN LIST
;
SETUNAD:    LDD   #UNADDR     ;ELSE SET UNADDRESS STATE
            STD   RVECT
;
UNADDR:     CLR   SCRTCH1     ;CLEAR SAIL MODE TO 00
            BCLR  SCCR2,$D2   ;TURN OFF XMIT INTERRUPTS
;&          LDAA  SCCR2
;           ANDA  #$2C        ;TURN OFF XMIT INTERRUPTS
;           STAA  SCCR2
            LDD   #XPASS      ;SET XMIT VECTOR TO PASS
XSET:       STD   XVECT
            JMP   SCIXIT
;
NEXTAD:     INX               ;POINT NEXT ADD CHAR
            INC   SCRTCH1     ;(& INC COUNTER)
            TST   0,X         ;TEST IT,
            BEQ   GOTADDR     ; TERMINATOR?
            JMP   SCIXIT      ;ELSE SAME VECTOR
;
GOTADDR:    LDD   #CONCHAR    ;REC CONTROL CHAR NEXT..
            JMP   RSET
;
; TEST FOR BAUD CHANGE COMMAND
TSTBAUD:    LDD   #BAUD2
            JMP   RSET
;
BAUD2:      CMPA  #'0'        ;GOT SECOND 0?
            BEQ   BAUD3
            JMP   SETUNAD
;
; COME HERE IF GOT #00 (BAUD CHANGE)
BAUD3:      LDX   #SCRTCH1    ;POINT TO 5 CHAR BUFFER
            STX   VECT2       ;SAVE AT VECT2
            LDD   #BAUD4
            JMP   RSET
;
BAUD4:      LDX   VECT2       ;GET POINTER
            CMPA  #' '        ;IS IT SPACE?
```

```
                BNE     BAUD5
                LDAA    #'0'            ;..CHANGE SPACES TO 0'S
BAUD5:          STAA    0,X             ;STORE CHAR
                INX                     ;ADVANCE POINTER
                CPX     #SCRTCH1+5      ;PAST END OF 5 CHAR BUFFER?
                BEQ     BAUD6
                STX     VECT2           ;IF NOT DONE, SAVE POINTER
                JMP     SCIXIT          ;AND GET MORE
;
BAUD6:          LDX     SCRTCH1+3       ;GET LAST TWO CHARS
                CPX     #$3030          ;ARE LAST TWO CHARS BOTH 0?
                BNE     WAITBRK         ;IF NOT VALID, SET UP TO WAIT FOR BREAK
;
; IF LAST TWO CHARS ARE 00, CHECK BUFFER AGAINST DATA TABLES FOR A MATCH
                LDD     #BAUDTBL        ;SET UP TABLE POINTER
                STD     VECT2           ;AND STORE IN JUMP2 VECTOR
;
;  TRY EACH TABLE SEPARATELY, 00 AS 1ST CHAR SIGNALS END OF TABLE
BTRY:           LDX     VECT2           ;GET TABLE POINTER
                LDAA    0,X             ; AND TABLE ELEMENT
                BEQ     WAITBRK         ; IF=00, WAS NO MATCH, WAIT FOR BREAK
;
;  COMPARE BUFFER WITH ONE TABLE ENTRY
                LDX     #SCRTCH1        ;INIT BUFFER POINTER TO TOP
                STX     XVECT           ; & SAVE IN XMIT JUMP VECTOR
;  COMPARE EACH CHARACTER
COMPARE:        LDX     XVECT           ;GET BUFFER POINTER
                LDAA    0,X             ; TO GET BUFFER CHAR TO A
                INX                     ; INX BUF POINT WHILE WE'RE HERE
                STX     XVECT           ;  & SAVE..
                LDX     VECT2           ;NOW GET TABLE POINTER
                CMPA    0,X             ; & USE TO COMPARE CHARS
                BNE     NOMATCH
CMATCH:         INX                     ;IF MATCH, MOVE AHEAD TABLE POINTER
                LDAA    0,X             ;LOOK AT NEXT TABLE ELEMENT
                BEQ     MATCH           ; IF IT'S 00, FULL MATCH!
                STX     VECT2           ; ELSE SAVE TABLE POINTER
                BRA     COMPARE         ; AND TRY NEXT CHAR
;
NOMATCH:        INX                     ;INC TABLE POINTER
                LDAA    0,X             ;AND LOOK FOR 00
                BNE     NOMATCH         ;LOOP TILL IT'S FOUND
                INX                     ;INC OVER NULL
                INX                     ; & BAUD SET BYTE
                STX     VECT2           ;AND REPLACE TO HOLDING LOCATION
                BRA     BTRY            ;GO BACK TO CHECK NEXT TABLE
;
WAITBRK:        LDAA    #$0C            ;TURN OFF SCI INTERRUPTS
                STAA    SCCR2           ; (REQUIRES TOF INTERRUPT TO RESTART)
                JMP     SETUNAD         ;AND SET VECTORS FOR NEW ADDRESS..
;
```

```
;*BAUD*  ALL LINES WITH THIS ITEM ARE MODIFIED FOR EACH BAUD GROUP
; MATCH TABLE FOR BAUD RATE SWITCH
BAUDTBL:    FCB   '096',$00,$30   ; 9600              *BAUD*
            FCB   '048',$00,$31   ; 4800 BAUD         *BAUD*
            FCB   '024',$00,$32   ; 2400              *BAUD*
            FCB   '012',$00,$33   ; 1200              *BAUD*
            FCB   '006',$00,$34   ;  600              *BAUD*
            FCB   '003',$00,$35   ;  300              *BAUD*
            FCB   $00             ;DONE
;
MATCH:      INX                   ;PASS 00 TERMINATOR
            LDAA  0,X             ;GET BAUD BYTE
            STAA  BAUD            ;AND RE-WRITE
            JMP   SETUNAD         ;AND PREPARE FOR NEW ADDRESSING...
;
;-------------------------------------------------------------
; ?M COMMAND - DATA INPUT SECTION
            if    VIEW
;.........................
QUERYM:     LDD   #SAVEADD        ;NOW INPUT ADDRESS
            JMP   IN4
;
SAVEADD:    LDD   SCRTCH1         ;GET ADDRESS,
            STD   SCRTCH3         ;& SAVE IN ADDRESS POINTER
            LDD   #STARTL
            JMP   IN4             ;THEN GET LENGTH
;
STARTL:     LDD   SCRTCH1         ;IF LENGTH NE 0000
            BNE   STARTL2         ; CARRY ON..
            JMP   PROMPT          ;ELSE QUIT
;
STARTL2:    LDD   #ECHO           ;PREPARE TO IGNORE ECHO CHARS
            STD   RVECT
            BSET  SCCR2,$E0       ;START XMIT INTERRUPTS
;&          LDAA  SCCR2
;           ORAA  #$E0            ;START XMIT INTERRUPTS
;           STAA  SCCR2
            JMP   OUTLINE         ;& SEND A LINE..
;
; ..THE ?M DUMP IS STORED IN THE XMIT SERVICE SECTION BELOW
            else
            endif
;
;------------------------------------------------------
; !M SECTION
            IF    LOAD            ;CONDITIONAL ASSEMBLY OF LOAD M SECTION
;.........................
BANGM:      LDD   #KEEPADD        ;INPUT STARTING ADDRESS
            JMP   IN4
;
KEEPADD:    LDD   #HIHEX          ;WAIT FOR HIGH HEX CHAR
```

```
                JMP     RSET
;
HIHEX:          JSR     ASCHEX          ;ATTEMPT CONVERT TO HEX
                BCS     TERMIN          ;CHECK TERMINATOR IF NOT..
                ASLA
                ASLA                    ;MOVE OVER NIBBLE
                ASLA
                ASLA
                STAA    SCRTCH5
                LDD     #LOWHEX
                JMP     RSET
;
LOWHEX:         JSR     ASCHEX          ;CONVERT LOW
                BCC     STASH
                JMP     PROMPT          ;PROMPT IF BAD SECOND CHAR
STASH:          LDX     SCRTCH1         ;GET POINTER
                ORAA    SCRTCH5         ;COMBINE NIBBLES
                STAA    0,X             ;STORE IT..
                INX                     ;INC &
                STX     SCRTCH1         ;RESTORE POINTER
                LDD     #HIHEX
                JMP     RSET
;
TERMIN:         ANDA    #$7F            ;MASK OFF FLAG BIT
                CMPA    #$0D            ;IS IT CR?
                BNE     TERMIN2
                JMP     PROMPT
;
TERMIN2:        CMPA    #' '            ;IS IT SPACE?
                BNE     TERMIN3
                LDD     #HIHEX
                JMP     RSET
;
TERMIN3:        CMPA    #';'            ;OR ';' ?
                BNE     TERMIN4
                LDD     #BANGM
                BRA     PASSLF
;
TERMIN4:        CMPA    #','            ;OR ',' ?
                BNE     TERMIN5
                LDD     #SETHIHX
PASSLF:         STD     VECT2
                LDD     #PASSLF2
                JMP     RSET
;
PASSLF2:CMPA            #$0A            ;GOT LF YET?
                BNE     PXIT
                JMP     JUMP2           ;IF YES, EXIT THRU JMP2
PXIT:           JMP     SCIXIT          ;ELSE WAIT FOR MORE CHARS..
;
SETHIHX:        LDD     #HIHEX          ;IF ',' THEN PREP FOR MORE HEX
```

```
                JMP     RSET
;
TERMIN5:        JMP     PROMPT .     ;TO PROMPT IF NOT VALID CHAR
                endif
;--------------------------------------------------
; IN4 HEX DIGITS
;       INPUTS HEX DIGITS TILL TERMINATED BY A SPACE OR
;       BY A CR. (OR BY UNADDRESS). RESULT LEFT AT SCRTCH1,2
;       EXITS THRU JUMP2 WHEN DONE (TARGET IN D ON ENTRY)
;
IN4:            STD     VECT2        ;STORE EXIT VECTOR
                CLR     SCRTCH1      ;CLEAR INPUT BUFFER
                CLR     SCRTCH2
                LDD     #IN4HEX      ;POINT CHAR RECEIVER
                JMP     RSET
;
IN4HEX:         JSR     ASCHEX       ;CONVERT TO HEX
                BCS     CKEND        ;IF NOT HEX, CHECK FOR END
                PSHA                 ;SAVE..
                LDAA    SCRTCH1      ;HI TO A
                LDAB    SCRTCH2
                ASLD                 ;SHIFT OVER FOUR
                ASLD
                ASLD
                ASLD
                STAA    SCRTCH1      ;RETURN HIGH
                STAB    SCRTCH2      ;AND B
                PULA                 ;RETRIEVE NEW NIBBLE
                ORAA    SCRTCH2      ;ADD NEW NIBBLE
                STAA    SCRTCH2      ; & UPDATE
                JMP     SCIXIT
;
CKEND:          ANDA    #$7F         ;CLEAN OFF MS FLAG BIT
                CMPA    #' '         ;IS IT SPACE?
                BEQ     INEXIT
                CMPA    #$0D         ;IS IT CR?
                BEQ     INEXIT
                JMP     SCIXIT       ;ELSE, GET ANOTHER CHAR
;
INEXIT:         JMP     JUMP2        ;IF DONE, CARRY ON THRU VECTOR..
;
;
;--------------------------------------------------
; INPUT ONE HEX BYTE AND CONTINUE THRU JUMP2
INBYTE:         ASLA                 ;MOVE OVER
                ASLA
                ASLA
                ASLA
                STAA    SCRTCH5      ;SAVE
                LDD     #INBYTE2     ;POINT TO NEXT SEGMENT
                JMP     RSET
```

```
;
INBYTE2:        JSR     ASCHEX          ;CONVERT LOW NIBBLE
                BCC     SAVEDAT
                JMP     SETUNAD         ;UNADDRESS IF ERROR
;
SAVEDAT         :ORAA   SCRTCH5         ;COMBINE
                JMP     JUMP2           ;AND CARRY ON THRU VECTOR2
;
;------------------------------------------------------------------
; NOW CHECK THE TRANSMIT SITUATION
CHECKX:         BRCLR SCSR,$80,XERROR
;&              LDAA    SCSR            ;MAKE SURE XMIT BIT WAS SET
;               ANDA    #$80
;               BEQ     XERROR
                JMP     XJUMP           ;GO THRU XMIT VECTOR
;
; THESE SEGMENTS ARE ENTERED THRU XJUMP AND XVECTOR..
XERROR:         NOP                     ;TRIG LOC FOR DIAGNOSTICS
;
; XJUMP TARGETS..
XPASS:          BCLR    SCCR2,$D0
;&              LDAA SCCR2
;               ANDA #$2F
;                STAA SCCR2
                JMP     SCIXIT
;
                if      VIEW            ;a conditional segment
; OUTPUT A LINE OF DATA (FROM ?M DUMP)
OUTLINE:        LDD     #OUTLF          ;& SET UP TO XMIT
                STD     XVECT
                LDAA    #$0D            ;SEND CR
                JMP     SETOUT
;
OUTLF:          LDD     #OUTADDR
                STD     XVECT
                LDAA    #$0A            ;SEND LF
                JMP     SETOUT
;
OUTADDR:        LDD     #LOWADDR        ;PREP OUTBYTE XIT
                STD     VECT2
                LDAA    SCRTCH3         ;GET HIGH ADDRESS
                JMP     OUTBYTE
;
LOWADDR:        LDD     #LOWADD2        ;EXIT FROM 2ND OUTBYTE..
                JMP     XSET
;
LOWADD2:        LDD     #ADDRSP
                STD     VECT2
                LDAA    SCRTCH4         ;NOW LOW ADDRESS
                JMP     OUTBYTE
;
```

```
ADDRSP:     LDD    #SENDSP      ;EXIT FROM 2ND OUTBYTE...
            JMP    XSET
;
SENDSP:     LDD    #OUTDATA
            STD    XVECT
            LDAA   #$20         ;SEND SPACE
            JMP    SETOUT
;
OUTDATA:    LDX    SCRTCH3      ;GET ADDR POINTER
            LDAA   0,X          ;TO GET DATA BYTE
            PSHA                ;SAVE
            LDD    #DECLEN
            STD    VECT2        ;PREP XIT VECTOR
            PULA                ;RETRIEVE DATA BYTE
            JMP    OUTBYTE      ;& SEND IT
;
DECLEN:     LDD    #DECLEN2     ;EXIT FROM OUTBYTE2..
            JMP    XSET
;
DECLEN2:    LDX    SCRTCH1      ;GET LENGTH
            DEX                 ;DECREMENT
            STX    SCRTCH1      ;& RETURN
            BNE    SAYMORE      ; MORE DATA?
            JMP    PROMPT       ;PROMPT IF DONE
;
SAYMORE:    LDX    SCRTCH3      ;GET ADDR
            INX                 ;INCREMENT
            STX    SCRTCH3      ;REPLACE
            XGDX                ;X REG TO ACCD
            LSRD                ;LSB TO C
            BCS    OUTDATA      ;NEXT BYTE NOW IF ODD
            ANDB   #$07         ;CHECK FOR XXX0 (LINE END)
            BNE    SENDSP       ;IF ONLY EVEN, SEND SP
;
            LDAA   #';'         ;IF NEW LINE, SEND CONTINUATION
            JSR    GENPAR
            STAA   SCDR
            LDD    #OUTLINE     ;& PREP FOR NEW LINE..
            JMP    XSET
            endif
;
;-----------------------------------------------------------
; OUTPUTS A BYTE IN A AS TWO HEX CHARS, EXITS VIA VECT2..
OUTBYTE:    STAA   SCRTCH5      ;STASH FOR LATER
            LSRA
            LSRA                ;MOVE OVER
            LSRA
            LSRA
            JSR    HEXASC       ;CONVERT
            PSHA                ;AND SAVE..
            LDD    #OUTBYT2
```

```
                STD   XVECT
                PULA                    ;RECOVER CHARACTER
                JMP   SETOUT
;
OUTBYT2:        LDAA  SCRTCH5
                ANDA  #$0F              ;CLEAN UP
                JSR   HEXASC
                JMP   OUTJMP            ;CONVERT, SEND & JUMP THRU VECT2
;
;---------------------------------------------------------
; SEND A PROMPT FROM THE MONITOR, THEN WAIT FOR CHARS
PROMPT:         LDX   #PLIST
                JMP   TTY0
PLIST:          FCB   $0D,$0A,':',$EE,$90,SCRTCH1,$AA
                FDB   PJUMP
PJUMP:          LDAA  SCRTCH1
                JMP   CONCHAR
;---------------------------------------------------------
; INITIATE AN OUTPUT SEQUENCE
SETOUT:         JSR   GENPAR
                STAA  SCDR              ;SEND FIRST CHAR
                BSET  SCCR2,$C0
;&               LDAA       SCCR2
;                ORAA       #$C0   ;TURN ON INTERRUPTS IF REQ'D
;                STAA       SCCR2
                LDD   #ECHO             ;& PREPARE TO ECHO
                JMP   RSET
;---------------------------------------------------------
; SENDS ETX & CARRIES ON THRU VECT2
XETX:           LDAA  #$03              ;SEND ETX
OUTJMP:         JSR   GENPAR            ;MAY ENTER HERE...
                STAA  SCDR              ;DRAGON AGAIN?
                JMP   JUMP2             ;NOW THRU JUMP2
;
;
;*************************************************************
; SUBROUTINES USED BY PTR DRIVER......
;----------------
ATTN:           LDAA  PORTA             ;GET THE VALUE IN PORT A
                CMPA  #DB00             ;IS 0 DB SET?
                BNE   ATTN1             ;IF NO GO ON
                LDAA  #DB03             ;IF YES SET 3 DB
                STAA  PORTA
                BRA   ATTNF             ;EXIT
ATTN1:          CMPA  #DB03             ;IS 3 DB SET?
                BNE   ATTN2             ;IF NO GO ON
                LDAA  #DB09             ;IF YES SET 9 DB
                STAA  PORTA
                BRA   ATTNF             ;EXIT
ATTN2:          CMPA  #DB12             ;IS 12 DB SET?
                BNE   ATTN3             ;IF NO GO ON
```

```
            LDAA  #DB18      ;IF YES SET 18 DB
            STAA  PORTA
            BRA   ATTNF      ;EXIT
ATTN3:      CMPA  #DB18      ;IS 18 DB SET?
            BNE   ATTN4      ;IF NO GO ON
            LDAA  #DB24      ;IF YES SET 24 DB
            STAA  PORTA
            BRA   ATTNF      ;EXIT
ATTN4:      CMPA  #DB24      ;IS 24 DB SET?
            BNE   ATTN5      ;IF NO GO ON
            LDAA  #DB30      ;IF YES SET 30 DB
            STAA  PORTA
            BRA   ATTNF      ;EXIT
ATTN5:      CMPA  #DB30      ;IS 30 DB SET?
            BNE   ATTN6      ;IF NO GO ON
            BRA   ATTNF      ;IF YES CAN'T CHANGE ANYMORE
ATTN6:      LDAA  #DB00      ;IF NON OF THE ABOVE ARE SET, SET 0 DB
            STAA  PORTA
ATTNF       RTS

INC:        LDAA  PORTA      ;GET THE VALUE IN PORT A
            CMPA  #DB30      ;IS 30 DB SET?
            BNE   INC1       ;IF NO GO ON
            LDAA  #DB24      ;IF YES SET 24 DB
            STAA  PORTA
            BRA   INCF       ;EXIT
INC1:       CMPA  #DB24      ;IS 24 DB SET?
            BNE   INC2       ;IF NO GO ON
            LDAA  #DB18      ;IF YES SET 18 DB
            STAA  PORTA
            BRA   INCF       ;EXIT
INC2:       CMPA  #DB18      ;IS 18 DB SET?
            BNE   INC3       ;IF NO GO ON
            LDAA  #DB12      ;IF YES SET 12 DB
            STAA  PORTA
            BRA   INCF       ;EXIT
INC3:       CMPA  #DB12      ;IS 12 DB SET?
            BNE   INC4       ;IF NO GO ON
            LDAA  #DB09      ;IF YES SET 9 DB
            STAA  PORTA
            BRA   INCF
INC4:       CMPA  #DB09      ;IS 9 DB SET?
            BNE   INC5       ;IF NO GO ON
            LDAA  #DB03      ;IF YES SET 3 DB
            STAA  PORTA
            BRA   INCF
INC5:       LDAA  #DB00      ;IF NON OF THE ABOVE ARE SET, SET 0 DB
            STAA  PORTA
INCF        RTS

;*************************************************************
```

```
; SUBROUTINES USED BY SAIL DRIVER......
;-----------------
; Converts low nibble in A to ASCII char in A
HEXASC:     ANDA  #$0F          ;CLEAN IT UP
            ADDA  #$F6
            BCC   HNUMBER
            ADDA  #$07          ;CONVERT
HNUMBER:    ADDA  #$3A          ; TO ASCII
            RTS
;
;--------------------------------------------------
; Converts ASCII char in A to nibble in A if it is HEX & clears C
;       Otherwise, returns origional char with C bit set.
ASCHEX:     PSHA                ;SAVE CHAR
            SUBA  #$30
            BCS   NOTHEX
            SUBA  #$0A
            BGE   TSTALPH
            ADDA  #$0A          ;REPAIR NUMB
            BRA   AXIT
TSTALPH:    SUBA  #$07          ;41->00 ?
            BLT   NOTHEX
            SUBA  #$06
            BGE   NOTHEX
            ADDA  #$10          ;REPAIR ALPH
AXIT:       INS                 ;POP OFF SAVED CHAR
            CLC                 ;AND CLEAR CARRY BIT
            RTS
;
NOTHEX:     PULA                ;GET CHAR
            SEC                 ;SET CARRY
            RTS
;--------------------------------------------------
; GENERATE PARITY- Char is passed in and out thru A
GENPAR:     PSHB                ;SAVE SOME STUFF..
            PSHX
;           TAB                 ;COPY CHAR TO B
;           LDX   #PARITY       ;POINT TO PARITY TABLE TOP
;           ABX                 ;ADD OFFSET TO POINT X TO CHAR
;           ORAA  0,X           ;ADD PARITY BIT
            PULX                ;..RESTORE
            PULB
            RTS
;
; PARITY TABLE
PARITY:     FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
            FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
            FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
            FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
; XX20
            FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
```

```
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
; XX40
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
; XX60
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
; XX80
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
; XXA0
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
; XXC0
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;A
; XXE0
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;A
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;B
        FCB $00,$80,$80,$00,$80,$00,$00,$80     ;B
        FCB $80,$00,$00,$80,$00,$80,$80,$00     ;A
;
;-----------------------------------------------------------
; CHARACTER LOOKUP TABLE
CHARTBL: FCB $41,$42,$43,$44,$45,$46,$47,$00    ;00-07
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;08-0F
        FCB $49,$4A,$4B,$4C,$4D,$4E,$50,$00     ;10-17
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;18-1F
        FCB $51,$52,$53,$54,$55,$56,$57,$00     ;20-27
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;28-2F
        FCB $59,$5A,$5B,$5C,$5D,$5E,$5F,$75     ;30-37
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;38-3F
        FCB $61,$62,$63,$64,$65,$66,$68,$74     ;40-47
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;48-4F
        FCB $69,$6A,$6B,$6C,$6D,$6E,$70,$74     ;50-57
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;58-5F
        FCB $71,$72,$73,$74,$75,$76,$77,$72     ;60-67
        FCB $00,$00,$00,$00,$00,$00,$00,$00     ;68-6F
;****************************************************************
```

```
; HELP FILE DUMP
SAYHELP:    LDX   #HELPFIL
            BRA   TTY0
;-----------------------------------------------------------
; A TTY ROUTINE FOR SAIL68 **DO NOT DELETE! USED BY PROMPT ETC!**
;       -USES $FD BYTES
;       -TO ENTER FOR XMIT, LDX #table AND THEN JMP TTY0
;       -TO ENTER IN REC MODE LDX #table AND JMP TTYR0
;       -MUST USE EE TO SWITCH FROM SENDING TO REC CHARS
;       -CAN GO BACK TO SENDING W/O PROBLEM
;       -ARG TABLE CODES..
;             ASCII -TYPE IT AND GO ON WITH XMIT INTERRUPTS ON
;             00    -BACK TO THE PROMPT
;             AA aaaa     -JMP TO aaaa WITH XMIT INTERRUPTS STILL ON (USE FDB!)
;             80 aa -SEND CHAR AT LOC aa
;             81 aa -SEND HEX PAIR AT LOC aa
;             EE    -SEND ETX & WAIT FOR A CHAR (MUST BE FOLLOWED BY 9X)
;             90 aa -REC A CHAR AND SAVE IT AT aa
;             91 aa -REC A HEX PAIR AND SAVE IT AT aa (ABORT TO PROMPT ->HEX)
;             94 aa -CLEAR SCRTCH1,2 & WAIT FOR A HEX STRING, PUT AT aa,aa+1
;                    (ENTRY TERMINATED BY SP OR CR)
;
TTY:        LDX   XTTY          ;LOAD X FROM STORAGE LOCATION
TTY0:       LDD   #TTY          ;SET REENTER VECTOR
            STD   XVECT
            LDAA  00,X          ;GET ARGUMENT
            BNE   TRYAA
            JMP   PROMPT                ;IF =00, GO DIRECTLY TO PROMPT..
;
TRYAA:      CMPA  #$AA
            BNE   TRYASC
            INX
            LDX   0,X           ;VECTOR TO X
            JMP   0,X           ;AND JUMP TO M(X)
;
TRYASC:     TSTA
            BMI   TRY80         ;IF MSB IS 1, NOT ASCII..
            JSR   LBISX         ;ELSE GET THE ASCII IMM CHAR
            TBA
            JMP   SETOUT        ;GO SEND IT NOW
;
TRY80:      CMPA  #$80          ;REQ IND CHAR?
            BNE   TRY81
            INX
            JSR   LBISX         ;GET THE ADDRESS (ASSUME BASE PAGE)
            JSR   CRADDR        ;CREATE ADDRESS
            LDAA  0,X           ;GET THE IND DATA
            JMP   SETOUT
;
TRY81:      CMPA  #$81          ;REQ IND HEX PAIR?
            BNE   TRYEE
```

```
                INX
                JSR     LBISX           ;GET THE ADDRESS (ASSUME BASE PAGE)
                JSR     CRADDR          ;CREATE ADDRESS
                LDD     #TTY2           ;THEN SET UP VECT2 EXIT PATH
                STD     VECT2
                LDAA    0,X             ;GET THE IND DATA NOW
                JMP     OUTBYTE
;
TTY2:           LDD     #TTY            ;EXIT PATH FOR VECT2 STUFF..
                JMP     XSET            ;(ALLOWS OUTBYTE TO RETURN TO TTY)
;
TRYEE:          CMPA    #$EE            ;SEND ETX & GO TO RECEIVE MODE?
                BEQ     GOTEE
TRERR:          JMP     PROMPT          ;IF NOT VALID OUT GROUP CONTROL CHAR, JUST
PROMPT
;
GOTEE:          INX                     ;SWITCH TO INPUT MODE
                STX     XTTY            ;SAVE THE CONTROL TABLE POINTER
                LDAA    #$03            ;SEND ETX
                JSR     GENPAR
                STAA    SCDR
                BCLR    SCCR2,$C0       ;STOP FURTHER XMIT INTERRUPTS
                LDD     #TTY3           ; AND PREPARE TO RECEIVE THE ETX ECHO
                JMP     RSET
;
TTY3:           CMPA    #$03            ;GOT ETX YET?
                BEQ     TTY4            ; KEEP WAITING TILL IT ECHOS
                JMP     SCIXIT
;
TTY4:           LDD     #TTYR           ;WHEN GOT ETX, PREP TO REC DATA CHARS
                JMP     RSET            ; SET RVECT TO TTYR AND EXIT
;-----------------------
; TTY RECEIVE SECTION, ENTER HERE FROM COMMAND CHAIN W/O NEW CHAR
TTYR0:          STX     XTTY            ;JUST SAVE TABLE POINTER
                BRA     TTY4            ; AND GO GET A NEW CHAR

;  ENTER HERE FROM RVECT WITH DATA CHAR IN A
TTYR:           LDX     XTTY            ; WHILE WE FIGURE OUT WHAT TO DO WITH IT
                JSR     LBISX           ;GET THE CONTROL CHAR..
TRY90:          CMPB    #$90            ;SAVE WHOLE CHAR
                BNE     TRY91
SAVCHR:         JSR     LBISX           ;IF 90, SAVE WHOLE CHAR AT IMM ADDRESS
                JSR     CRADDR          ;CREATE ADDRESS
                STAA    0,X             ;SAVE THE DATA BYTE IN M(X)
                BRA     CKNEXT          ;CHECK NEXT CONTROL BYTE FOR TYPE (READ OR
WRITE)
;
TRY91:          CMPB    #$91            ;ENTER A HEX PAIR?
                BNE     TRY94
                JSR     ASCHEX
                BCS     TRERR           ;ABORT IF NOT HEX CHAR
```

```
            ASLA
            ASLA                 ;MOVE OVER
            ASLA
            ASLA
            STAA   SCRTCH1       ;SAVE TILL NEXT PASS
            LDD    #TTYHEX       ;PREP FOR 2ND HEX CHAR
            JMP    RSET
;
TTYHEX:     JSR    ASCHEX        ;( SECOND HALF OF TRY91 )
            BCS    TRERR         ;ABORT IF NOT A VALID HEX CHAR
            ORAA   SCRTCH1       ;COMBINE HIGH AND LOW CHARS
            LDX    XTTY
            JSR    LBISX         ;GET SINK ADDRESS
            JSR    CRADDR        ;CREATE ADDRESS
            STAA   0,X
            LDD    #TTYR         ;REPAIR THE VECTOR
            STD    RVECT         ; AND ON TO CHECK NEXT COMMAND
            BRA    CKNEXT
;
TRY94:      CMPB   #$94          ;IS IT IN4?
            BNE    TRERR         ;QUIT IF NOT A VALID CODE
GOT94:      CLR    SCRTCH1       ;GOT FIRST DIGIT IN A, CLEAR INPUT BUFFER
            CLR    SCRTCH2
            PSHA                 ;SAVE THE ARGUMENT
            LDD    #SAVE4
                   VECT2         ;PREPARE EXIT VECTOR
            LDD    #IN4HEX
            STD    RVECT         ;CHANGE RVECT
            PULA
            JMP    IN4HEX        ;AND PROCESS THE FIRST DIGIT
;
SAVE4:      LDD    #TTYR
            STD    RVECT         ;REPAIR THE VECTOR
            LDX    XTTY          ;GET THE TABLE POINTER
            JSR    LBISX         ;AND GET THE SINK ADDRESS
            JSR    CRADDR        ;CREATE ADDRESS
            LDD    SCRTCH1       ;AND STORE THE DOUBLE BYTE
            STD    0,X
            BRA    CKNEXT        ;CHECK NEXT CONTROL CHAR
;
CKNEXT:     LDX    XTTY          ;IS THE NEXT CONTROL CHAR AN INP OR OUT?
            LDAA   0,X           ;GET THE CONTROL CHAR
            ANDA   #$F0          ; AND CHECK THE HIGH NIBBLE
            CMPA   #$90          ;IS IT AN INPUT TYPE?
            BNE    TTYOL         ;IF NOT, SWITCH TO OUTPUT MODE
            JMP    SCIXIT        ;ELSE GET ANOTHER CHAR
;
TTYOL:      JMP    TTYO
;--
LBISX:      LDAB   0,X           ;SUBROUTINE TO LOAD B, INC & STR X
            INX                  ; USED BY TTY ABOVE
```

```
            STX     XTTY
            RTS
;--
CRADDR:     LDX     #$0000          ;SUBROUTINE TO CREATE AN ADDRESS
            ABX
            RTS
;-------------------------------------------------------
; AFTER VALID ADDRESS, RECEIVE CONTROL CHAR..
CONCHAR:    ANDA    #$7F            ;CLEAN OFF FLAG BIT
            CMPA    #'!'            ;IS IT '!'
            BEQ     SETBG
            CMPA    #'?'            ;OR IS IT '?'
            BEQ     SETQG
            CMPA    #'O'            ;IS IT 'O', START DATA COLLECTION?
            BEQ     TIMEJ
            CMPA    #'P'            ;IS IT 'P', PAUSE DATA COLLECTION?
            BEQ     TIMOUTJ
            CMPA    #'-'            ;IS IT '-', DATA SCAN REQUEST?
            BEQ     DSCANL
            CMPA    #'A'            ;IS IT'A', SET GAIN CONTROL?
            BEQ     ATTCMD
            CMPA    #' '            ;SPACE TO ENTER MONITOR?
            BEQ     GOMON
            CMPA    #$0D            ;CHAR RETURN TO ENTER MONITOR?
            BEQ     GOMON
            CMPA    #'H'            ;HELP?
            BEQ     SAYHEL
            CMPA    #'R'            ;RESTART?
            BEQ     RSTARTJ
            CMPA    #'V'            ;ATTEN VALUE REQUEST
            BEQ     AVOUTJ
            JMP     SETUNAD         ;UNADDRESS IF NOT VALID..
;
GOMON:      JMP     PROMPT          ;LINK INTO MONITOR
SAYHEL:     JMP     SAYHELP         ;LINK TO SAYHELP
DSCANL:     JMP     DSCAN           ;LINK TO DATA SCAN
TIMEJ:      JMP     TIME            ;LINK TO TIME
TIMOUTJ:    JMP     TIMOUT          ;LINK TO TIMOUT
RSTARTJ:    JMP     START           ;LINK TO START
AVOUTJ:     JMP     AVOUT           ;LINK TO ATTEN VAL OUTPUT
;-----------
SETBG:      LDD     #BGROUP         ;PREPARE FOR THE CHAR AFTER !
            JMP     RSET
SETQG:      LDD     #QGROUP         ;PREPARE FOR THE CHAR AFTER ?
            JMP     RSET
ATTCMD:     LDD     #ATTNGRP        ;PREPARE FOR THE CHAR AFTER A
            JMP     RSET
;-----------
BGROUP:     NOP
            CMPA    #'T'
            BEQ     LTESTJ
```

```
                if      LOAD
                 CMPA #'M'
                 BEQ    BANGML
                endif
                JMP     PROMPT
;
                if      LOAD
BANGML:          JMP    BANGM           ;LINK TO !M
                else
                endif

LTESTJ:         JMP     LTEST
;------------
QGROUP:         NOP
                if      VIEW
                CMPA    #'M'
                BEQ     QUERYML
                else
                endif
                JMP     PROMPT
;
                if      VIEW
QUERYML:        JMP     QUERYM          ;LINK TO ?M
                else
                endif
;----------------------
ATTNGRP:        NOP
                CMPA    #'0'            ;MANUAL SET 0 DB ATTEN
                BEQ     SET0DBJ
                CMPA    #'1'            ;MANUAL SET 3 DB ATTEN
                BEQ     SET1DBJ
                CMPA    #'2'            ;MANUAL SET 9 DB ATTEN
                BEQ     SET2DBJ
                CMPA    #'3'            ;MANUAL SET 12 DB ATTEN
                BEQ     SET3DBJ
                CMPA    #'4'            ;MANUAL SET 18 DB ATTEN
                BEQ     SET4DBJ
                CMPA    #'5'            ;MANUAL SET 24 DB ATTEN
                BEQ     SET5DBJ
                CMPA    #'6'            ;MANUAL SET 30 DB ATTEN
                BEQ     SET6DBJ
                CMPA    #'A'            ;SET AGC ON
                BEQ     AGCONJ
                JMP     PROMPT

SET0DBJ:        JMP     SET0DB          ;LINK TO SET0DB
SET1DBJ:        JMP     SET03DB         ;LINK TO SET3DB
SET2DBJ:        JMP     SET09DB         ;LINK TO SET09DB
SET3DBJ:        JMP     SET12DB         ;LINK TO SET12DB
SET4DBJ:        JMP     SET18DB         ;LINK TO SET18DB
SET5DBJ:        JMP     SET24DB         ;LINK TO SET24DB
```

```
SET6DBJ:    JMP   SET30DB      ;LINK TO SET30DB
AGCONJ:     JMP   AGCON        ;LINK TO AGCON

;-----------------------------------------------------------
;PAUSE REQUEST                  STOPS THE DATA DUMP
TIMOUT:     CLR   TMSK1        ;DISABLE TOC2 INTERUPT
            LDAA  #$FF         ;RESET TOC FLAGS
            STAA  TFLG1
            JMP   PROMPT       ;RETURN
;-----------------------------------------------------------
;START REQUEST                  TURN ON TIMER INTERUPT
TIME:       LDD   TCNT         ;LOAD THE CURRENT TIMER VALUE
            ADDD  #SAMPLE      ;SET AN INITIAL DELAY OF 1.0mSEC
            STD   DTOC2
            LDAA  #$40         ;TURN ON OC2 INTERUPT
            STAA  TMSK1
            JMP   PROMPT       ;RETURN
;-----------------------------------------------------------
; DATA SCAN
DSCAN:      LDX   #DSCAN0
            JMP   TTY0
DSCAN0:     DB    $AA
            DW    DSCAN1
DSCAN1:     BRCLR TX1FLG,$01,DSCAN2   ;HAS BUFFER 1 BEEN SENT YET?
            BRCLR TX2FLG,$01,DSCAN3   ;HAS BUFFER 2 BEEN SENT YET?
NOTYET:     LDX   #NOTRDY      ;WAIT UNTIL A BUFFER IS READY
            JMP   TTY0
NOTRDY:     FCB   $00,$00
;
DSCAN2:     BSET  TX1FLG,$01   ;SET BUFFER 1 TX'D FLAG
            LDX   #MAXBUF1
            BRA   DSCAN4
DSCAN3:     BSET  TX2FLG,$01   ;SET BUFFER 2 TX'D FLAG
            LDX   #MAXBUF2
DSCAN4:     STX   TXBUFF       ;SAVE THE BUFFER TO TX IN TXBUFF
            LDAB  MAXNUM
            ABX
            STX   TXBFULL      ;GENERATE THE BUFFER DONE VALUE
;
            LDY   TXBUFF
            STY   TXINT
DSCAN5:     LDAA  $00,Y        ;GET THE DATA POINT HI BYTE
            STAA  TXMHI        ;STORE IN BASE PAGE
            LDAA  $01,Y        ;GET THE DATA POINT LOW BYTE
            STAA  TXMLO        ;STORE IN BASE PAGE
            LDAA  $10,Y        ;GET THE INDEX HI BYTE
            STAA  TXIHI
            LDAA  $11,Y        ;GET THE INDEX LOW BYTE
            STAA  TXILO
;
            LDX   #BITSND1     ;PREP FOR TX
```

```
                JMP     TTY0            ;TX DATA POINT
BITSND1:        DB      $81,TXMHI,$AA ;TX VIA TTY
                DW      DTX1
DTX1:           LDX     #BITSND2        ;PREP FOR TX
                JMP     TTY0            ;TX DATA POINT
BITSND2:        DB      $81,TXMLO,$AA ;TX VIA TTY
                DW      DTX2
DTX2:           LDX     #BITSND3        ;PREP FOR TX
                JMP     TTY0            ;TX DATA POINT
BITSND3:        DB      $81,TXIHI,$AA ;TX VIA TTY
                DW      DTX3
DTX3:           LDX     #BITSND4        ;PREP FOR TX
                JMP     TTY0            ;TX DATA POINT
BITSND4:        DB      $81,TXILO,$AA ;TX VIA TTY
                DW      DSCAN6
;
DSCAN6:         LDY     TXINT
                INY
                INY
                STY     TXINT
                CPY     TXBFULL         ;HAS THE ENTIRE BUFFER BEEN SENT?
                BLO     DSCAN5
;
                LDAA    #$00
                LDX     TXBUFF          ;CLEAR THE BUFFER THAT WAS TX'D
DSCAN7:         STAA    $00,X
                INX
                CPX     TXBFULL
                BLO     DSCAN7
;
                LDX     #DSCAN8         ;END TX
                JMP     TTY0
DSCAN8:         DB      $00,$00
;-------
LTEST:          LDX     #RTEST
                JMP     TTYR0
;
RTEST:          DB      $94,$80," GOT 4! SEND 4 MORE",$EE,$94,$84,"DONE",$00,$00
;
;-----------------------------------------------------
;0 DB MANUAL GAIN REQUEST
SET0DB:         BSET    AGCFLG,$01      ;TURN OFF AGC
                LDAA    #DB00           ;SET 0 DB ATTENUATION
                STAA    PORTA
                JMP     PROMPT          ;RETURN
;-----------------------------------------------------
;3 DB MANUAL GAIN REQUEST
SET03DB:        BSET    AGCFLG,$01      ;TURN OFF AGC
                LDAA    #DB03           ;SET 3 DB ATTENUATION
                STAA    PORTA
                JMP     PROMPT          ;RETURN
```

```
;-----------------------------------------------------
;9 DB MANUAL GAIN REQUEST
SET09DB:      BSET  AGCFLG,$01    ;TURN OFF AGC
              LDAA  #DB09         ;SET 9 DB ATTENUATION
              STAA  PORTA
              JMP   PROMPT        ;RETURN
;-----------------------------------------------------
;12 DB MANUAL GAIN REQUEST
SET12DB:      BSET  AGCFLG,$01    ;TURN OFF AGC
              LDAA  #DB12         ;SET 12 DB ATTENUATION
              STAA  PORTA
              JMP   PROMPT        ;RETURN
;-----------------------------------------------------
;18 DB MANUAL GAIN REQUEST
SET18DB:      BSET  AGCFLG,$01    ;TURN OFF AGC
              LDAA  #DB18         ;SET 18 DB ATTENUATION
              STAA  PORTA
              JMP   PROMPT        ;RETURN
;-----------------------------------------------------
;24 DB MANUAL GAIN REQUEST
SET24DB:      BSET  AGCFLG,$01    ;TURN OFF AGC
              LDAA  #DB24         ;SET 24 DB ATTENUATION
              STAA  PORTA
              JMP   PROMPT        ;RETURN
;-----------------------------------------------------
;30 DB MANUAL GAIN REQUEST
SET30DB:      BSET  AGCFLG,$01    ;TURN OFF AGC
              LDAA  #DB30         ;SET 30 DB ATTENUATION
              STAA  PORTA
              JMP   PROMPT                ;RETURN
;-----------------------------------------------------
;AGC ON REQUEST
AGCON:        CLR   AGCFLG        ;SET AGC ON FLAG TO 0 (TURN AGC ON)
              JMP   PROMPT                ;RETURN
;-----------------------------------------------------
AVOUT:        LDX   #AVODAT
              JMP   TTY0
AVODAT:       DB    $81,PORTA,$00,$00
;             JMP   PROMPT
;*********************************************************
; HELP FILE
HELPFIL       :FCB  'elp SAIL DRIVER Mon, Jun 7, 1993, 11:53',$0D,$0A
              FCB   'Monitor Commands (enter via _ or ? or !)',$0D,$0A
              FCB   ' _ (space) enter monitor*',$0D,$0A
              FCB   ' ?Maaaa_llllcr *',$0D,$0A
              FCB   ' !Maaaa_dd...cr *',$0D,$0A
              FCB   'Attenuator Commands (enter via A)',$0D,$0A
              FCB   ' A0 sets  0 dB manual attenuation',$0D,$0A
              FCB   ' A1 sets  3 dB manual attenuation',$0D,$0A
              FCB   ' A2 sets  9 dB manual attenuation',$0D,$0A
              FCB   ' A3 sets 12 dB manual attenuation',$0D,$0A
```

```
        FCB     ' A4 sets 18 dB manual attenuation',$0D,$0A
        FCB     ' A5 sets 24 dB manual attenuation',$0D,$0A
        FCB     ' A6 sets 30 dB manual attenuation',$0D,$0A
        FCB     ' AA turns AGC on',$0D,$0A
        FCB     'O starts receiver collecting samples',$0D,$0A
        FCB     'P pauses receiver',$0D,$0A
        FCB     'R resets the system',$0D,$0A
        FCB     'with all RCA UT4 conventions',$0D,$0A
        FCB     ' * returns prompt',$00
;***************************************************************
        ORG $FFD6
VSCI:   FDB     SCI
VSPI:   FDB     START
VPAIE:  FDB     START
VPAO:   FDB     START
VTOF:   FDB     START
VTOC5:  FDB     START
VTOC4:  FDB     START
VTOC3:  FDB     START
VTOC2:  FDB     OC2
VTOC1:  FDB     START
VTIC3:  FDB     START
VTIC2:  FDB     START
VTIC1:  FDB     START
VRTI:   FDB     START
VIRQ:   FDB     START
VXIRQ:  FDB     START
VSWI:   FDB     START
VILLOP: FDB     START
VCOP:   FDB     START
VCLM:   FDB     START
VRST:   FDB     START
        END
```

Appendix 3

This appendix contains the C source code use by the Macintosh computer for the algorithm of figure 2-4. The user interface follows the guidelines specified by Apple, and the Event Manager is used to initiate all functions. After the hardware is initialized, the program waits in the EventLoop until an event is received. If no events are received in the loop cycle time, the program requests data from the 68HC11 and the GPS receiver. If the data is available, it is stored to form experiment vectors. Once enough experiment vectors are collected, the maximum likelihood estimate is calculated. The program the returns to the EventLoop.

```
/*************************************************************************

TRACKER.C

Dennis Michael Wojcik
15 July 1993

This program is version 1.0 of the Precision Tracking Receiver. It is tailored
for the scenario of a stationary acoustic beacon and a moving observer. The
program assumes that the observer has a GPS receiver for positioning data
that can produce a fix at least every 2 seconds. If a different navigation
system is used to provide ship locations, it is up to the user to provide
the proper position format:

        ?L HHMMSS,DDMM.hh,N,DDDMM.hh,E,x,x,DOP,xxx,M,xxxxx,M

*************************************************************************/

/************************** INCLUDES ****************************************/

#include <math.h>               // needed for math calls
#include <AppleEvents.h>        // needed to handle AppleEvents
#include <GestaltEqu.h>         // needed to read system 7 AppleEvents
#include <Values.h>             // contains needed Mac constants
#include <serial.h>             // has needed serial comms constants
#include <Packages.h>           // needed to call time functions
#include <string.h>             // needed to call string functions
#include <float.h>              // contains needed constants
#include <stdlib.h>             // for standard calls such as strtoul etc
#include <stdio.h>              // usefull for debugging
/*************************************************************************/
```

```
/************************** DEFINES *****************************************/

#define  kBaseResID           128                   // define general resource infor
#define kMoveToFront          (WindowPtr)-1L
#define kUseDefaultProc       (void *)-1L
#define kSleep                1L
#define kLeaveWhereItIs       false

#define mApple                kBaseResID            // Apple menu resource ID
#define iAbout                1

#define mFile                 kBaseResID+1          // File menu resource ID
#define iQuit                 1

#define  mAttenuator          kBaseResID+3          // Attenuator menu resource ID
#define iAuto                 1
#define i0dB                  2
#define i3dB                  3
#define i9dB                  4
#define i12dB                 5
#define i18dB                 6
#define i24dB                 7
#define i30dB                 8

#define mBaud                 kBaseResID+4          // Baud Rate menu resource ID
#define i300                  1
#define i600                  2
#define i1200                 3
#define i2400                 4
#define i4800                 5
#define i9600                 6

#define mSignal               kBaseResID+5          // Signal menu resource ID
#define iPeriod               1
#define iNumMax                      2
#define iWindow                      3
#define iMessageText          4
#define kPeriodDialog         kBaseResID
#define kNumMaxDialog         kBaseResID+1
#define kWindowDialog         kBaseResID+2

#define kLeftMargin           4                     // define screen margins
#define kFontSize             10
#define kRowHeight            (kFontSize+2)
#define kDataScanHeight       1
#define kHorizontalOffset     0
#define kGestaltMask          1L
#define kHorizontalPix        30
#define kVerticalPix          40
#define kMinWindowHeight      50
#define kMinWindowWidth       80
#define kMinTextPosition      0
```

```
#define kMaxTextPosition        32767

#define kScrollbarAdjust        (16-1)          // define update constants
#define kNormalUpdates          true

#define bufferSize              1024            // misc. default values
#define k16Bit                  4
#define k8Bit                   2
#define kHexBase                16
#define kpi                     3.141592654

#define kNumExp                 6               // default experiment parameters
#define kParameters             3
#define kNumMeas                6
#define kStep                   15
#define kC                      1500
#define kMaxR                   5000.0
#define kMaxD                   -5000.0
#define kMinD                   -1.0
#define kxvar                   9.0
#define kyvar                   9.0
#define ktvar                   8.3333e-2
/****************************************************************************/

/*************************** MACROS *******************************************/

#define TopLeft( r )            (*(Point *) &(r).top)
#define BottomRight( r )        (*(Point *) &(r).bottom)
/****************************************************************************/

/************************** GLOBALS ******************************************/

int             gASerRefIn, gASerRefOut;
Boolean         gDone, gHasPopupControl, gDataScan;
Rect            gTrackerRect;
int             gverticalPixels;
int             gRowStart, gDataRowStart, gserConfig;
unsigned long   gPeriod=2000UL;
long            gNumMax=2L, gWindow=20L;
char            gDataBuffer[bufferSize];
char            gGPSBuffer[bufferSize]=
                    "L 145642,0000.00,N,00000.00,E,0,0,000,000,M,0000,M";
short           gLastAttn=1, gLastBaud=6;
Str255          gItemText;
unsigned long   ghorizontalPixels;
WindowPtr       gWetPaperWind, gTerminalWind, gRangeWind;

long            gPulseCounter=0L;
long            gCheckVal=1L;
unsigned long   gtime1, gtime2;
double          gXref, gYref, gX1, gX2, gY1, gY2;
Boolean         gstarter=true;
```

```
Boolean         greflag=true;
Boolean         gDataWrap=false, trackerStart=false;
double          gYFact;
unsigned long   gt_mu1, gt_mu2;
double          gV[kNumMeas]={kxvar,kyvar,ktvar,kxvar,kyvar,ktvar};
double          gtheta[3]={ 0.0, 0.0, -2000.0}, gPosition[kParameters];
double          *gw;

double          gxold=0.0, gyold=0.0;
int             gnumIts=0;
int             gmu=0;
long            gi=0L,gnumExp=0L;
double          gj=1.0;
char            gYrefString[8];
char            gXrefString[9];
/****************************************************************************/

/************************* FUNCTION PROTOTYPES*******************************/

void            ToolBoxInit(void);
void            WindowInit(void);
void            MenuBarInit(void);
void            SerialInit(void);
void            EventLoop(void);
void            DoEvent( EventRecord *eventPtr );
unsigned long   HandleDataIn(int numDataPoins);
void            HandleMouseDown(EventRecord *eventPtr);
void            HandleMouseUp(EventRecord *eventPtr);
void            HandleMenuChoice( long menuChoice );
void            HandleAppleChoice( short item );
void            HandleFileChoice( short item );
void            HandleAttenuator(short item);
void            HandleBaud(short item);
void            HandleSignal(short item);
unsigned long   HandleDialog(int dialogRes);
void            ConcatString(Str255 str1, Str255 str2);
void            HandleKeyIn( char theChar);
int             TalkSail( char theChar);
void            DisplayString(int numDataPoints);
void            ScrollDisplay(void);
void            HandleNull( void );
unsigned        HexCharToInt(char Bite,int exponent);
int             DecIntToHexChar(int decimal);
void            DoUpdate(EventRecord *eventPtr);
int             Power(int base, int exponent);
int             DecIntToHexChar(int decimal);
void            ScrollDataScan(void);
unsigned long   DecTimeToSec(unsigned long  timeVal);
double          LatToY(double latVal);
double          LongToX(double longVal, double cosfact);
void            tracker(long numExp);
pascal OSErr    SetDialogDefaultItem(DialogPtr theDialog, short newItem)
```

```
                        ={ 0x303C, 0x0304, 0xAA68};
pascal OSErr    SetDialogCancelItem(DialogPtr theDialog, short newItem)
                        ={0x303C, 0x0305, 0xAA68};
pascal OSErr    SetDialogTracksCursor(DialogPtr theDialog, Boolean tracks)
                        ={0x303C, 0x0306, 0xAA68};
/**************************************************************************/

/*--------------------------Main-------------------------------------*/
void    main(void)
{
        WindowPtr       window;
        OSErr           RAMOpenErr, resetErrIn, resetErrOut, bufSizeErr;
        int             SerConfig;
        Ptr             serBPtr;
        char            serInBuffer[bufferSize];
        long            numCharIn, numCharOut, maxtime;
        char            addressBuf[20];

        ToolBoxInit();                                  // initialize the toolboxes
        WindowInit();                                   // set up the windows
        MenuBarInit();                                  // draw the menu bar
        SerialInit();                                   // open the serial drivers

        window=FrontWindow();                           // ensure we're in the window
        RAMOpenErr=RAMSDOpen(sPortA);                   // open the RAM Serial Drivers
                if (RAMOpenErr != noErr)                // if we can't open RAM Driver beep
                {
                        SysBeep(10);
                        ScrollDisplay();
                        DrawString("\pProblem opening the RAM Serial Drivers.");
                }
        gserConfig=baud9600+data8+stop10+noParity;      // Set the protocal for 9600
                                                        // 8 data, 1 stop, no parity
        resetErrIn=SerReset(gASerRefIn,gserConfig);
        resetErrOut=SerReset(gASerRefOut,gserConfig);
                if ((resetErrIn != noErr) || (resetErrOut != noErr))
                {
                        SysBeep(10);                    // if we can't set the protocal
                        ScrollDisplay();                // beep
                        DrawString("\pProblem setting the serial protocal.");
                }

        serBPtr=serInBuffer;                            // Set the serial buffer size
        bufSizeErr=SerSetBuf(gASerRefIn,serBPtr,bufferSize);
                if (bufSizeErr != noErr)
                {
                        SysBeep(10);                    // if we can't set the buffer
                        ScrollDisplay();                // size, beep
                        DrawString("\pProblem resizing the serial input buffer.");
                }

        ScrollDisplay();
```

```
numCharOut=5L;                                  // see if we can talk to the GPS receiver
FSWrite(gASerRefOut, &numCharOut, "#GPS ");
SerGetBuf(gASerRefIn,&numCharIn);
maxtime=TickCount()+20L;
while(numCharIn < 9L)
{
        SerGetBuf(gASerRefIn,&numCharIn);
        if(TickCount() > maxtime)
        {
                SysBeep(10);
                DrawString("\p Can't talk to the GPS reciever, unable to run!!");
                ScrollDisplay();
                break;
        }
}
FSRead(gASerRefIn,&numCharIn,&addressBuf);

numCharOut=5L;                                  // see if we can talk to the PTR receiver
FSWrite(gASerRefOut, &numCharOut, "#PTR ");
SerGetBuf(gASerRefIn,&numCharIn);
maxtime=TickCount()+20L;
while(numCharIn < 9L)
{
        SerGetBuf(gASerRefIn,&numCharIn);
        if(TickCount() > maxtime)
        {
                SysBeep(10);
                DrawString("\p Can't talk to the PTR reciever, unable to run!!");
                ScrollDisplay();
                maxtime=TickCount()+120L;
                while(TickCount() < maxtime)
                        TickCount();
                abort();
        }
}
FSRead(gASerRefIn,&numCharIn,&addressBuf);

TalkSail('O');                                  // start the PTR's data collection

DrawString("\p: ");                             // display a cursor

gw = calloc(3L*kStep*(kNumExp+1),sizeof(double));       // get memory for data
if (gw == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}

EventLoop();                            // main execution loop

RAMSDClose(sPortA);                     // must close the RAM Drivers before shutdown
```

```
	free(gw);				// must release the memory before shutdown
}
/*---------------------------------------------------------------------*/


/*------------------------ToolBoxInit--------------------------------*/
void	ToolBoxInit(void)
{								// initialize all of the ToolBox
	InitGraf(&thePort);				// Routines
	InitFonts();
	InitWindows();
	InitMenus();
	TEInit();
	InitDialogs(nil);
	InitCursor();
}
/*---------------------------------------------------------------------*/


/*-------------------------WindowInit---------------------------------*/
void	WindowInit(void)
{								// initialize the display windows
	WindowPtr		window;
	short			fontNum;
	Boolean		update = true;

	window = GetNewWindow(kBaseResID, nil, kMoveToFront);

	if (window == nil)
	{
		SysBeep(10);				// Couldn't load the WIND resource
		ExitToShell();
	}

	SetPort(window);				// set the window to be active

	GetFNum("\pTimes", &fontNum);		// use Times font
	if (fontNum != 0) TextFont(fontNum);
	TextSize(kFontSize);				// use the default font size

	gverticalPixels=screenBits.bounds.bottom-screenBits.bounds.top-45;
	ghorizontalPixels=screenBits.bounds.right-screenBits.bounds.left-10;
	gDataRowStart=gverticalPixels-2*kFontSize;

	SizeWindow(window, ghorizontalPixels, gverticalPixels, update);

	ShowWindow(window);				// activate the window
	gWetPaperWind=FrontWindow();

	gTerminalWind = GetNewWindow(kBaseResID+1, nil, kMoveToFront);

	if (gTerminalWind == nil)
```

```
	{
		SysBeep(10);				// Couldn't load the WIND resource
		ExitToShell();
	}

	SetPort(gTerminalWind);			// set the window to be active

	GetFNum("\pTimes", &fontNum);		// use Times font
	if (fontNum != 0) TextFont(fontNum);
	TextSize(kFontSize);			// use the default font size
	gRowStart=gTerminalWind->portRect.bottom-2*kFontSize;

	ShowWindow(gTerminalWind);
}
/*-------------------------------------------------------------------*/


/*------------------------MenuBarInit----------------------------------*/
void	MenuBarInit(void)
{
	Handle		menuBar;
	MenuHandle	menu;

	menuBar = GetNewMBar( kBaseResID );
	SetMenuBar( menuBar );

	menu = GetMHandle( mApple );
	AddResMenu( menu, 'DRVR');

	DrawMenuBar();
}
/*-------------------------------------------------------------------*/


/*------------------------SerialInit---------------------------------*/
void	SerialInit(void)
{
	WindowPtr	window;
	OSErr		openErr, RAMErr, resetErr ;	// define system errors
	int		serConfig;			// define the port configuration
	char		driverStringIn[]="\p.AIn";	// define the input driver name
	char		driverStringout[]="\p.Aout";	// define the output driver name

	window=FrontWindow();

	openErr=OpenDriver(driverStringIn,&gASerRefIn); // Open the port A ROM input serial driver
		if (openErr != 0)
		{
			SysBeep(10);
			ScrollDisplay();
			DrawString("\pCan't open the ROM Serial input driver");
			return;
```

```
		}

	openErr=OpenDriver(driverStringout,&gASerRefOut);	// Open the port A ROM output
								//serial driver
		if (openErr != 0)
		{
			SysBeep(10);
			ScrollDisplay();
			DrawString("\pCan't open the ROM Serial output driver");
			return;
		}

	gserConfig=baud9600+data8+stop10+noParity;	// Set the protocal for
							// 9600, 7D, 1S, no parity

	resetErr=SerReset(gASerRefIn,gserConfig);	// Set the input driver to the new protocal
		if (resetErr != 0)
		{
			SysBeep(10);
			ScrollDisplay();
			DrawString("\pCan't set the ROM Serial input protocal");
			return;
		}

	resetErr=SerReset(gASerRefOut,gserConfig);	// Set the output driver to the new protocal
		if (resetErr != 0)
		{
			SysBeep(10);
			ScrollDisplay();
			DrawString("\pCan't set the ROM Serial output driver");
			return;
		}

}
/*-----------------------------------------------------------------*/


/*----------------------EventLoop----------------------------------*/
void	EventLoop(void)
{
	EventRecord	event;

	gDone = false;
	while (gDone == false)				// loop here until done with program
	{
		if (WaitNextEvent(everyEvent, &event, kSleep, nil))
			DoEvent(&event);
		else
			HandleNull();			// collect data here
	}
}
/*-----------------------------------------------------------------*/
```

```
/*--------------------------DoEvent-------------------------------------*/
void	DoEvent(EventRecord *eventPtr)
{
	Boolean		becomingActive;
	char		theChar;

	switch (eventPtr->what)
	{
		case mouseDown:
			HandleMouseDown(eventPtr);
			break;
		case mouseUp:
			HandleMouseUp(eventPtr);
			break;
		case keyDown:
			theChar = eventPtr->message & charCodeMask;
			if((eventPtr->modifiers & cmdKey) != 0)
				HandleMenuChoice(MenuKey( theChar));
			else
				HandleKeyIn( theChar );
			break;
		case updateEvt:
			DoUpdate( eventPtr);
			break;
/*		case activateEvt:
			becomingActive = ((eventPtr->modifiers &
								activeFlag) ==
								activeFlag);
			DoActivate((WindowPtr)eventPtr->message,
							becomingActive);
			break;*/
	}
}
/*-----------------------------------------------------------------*/


/*----------------------HandleNull--------------------------------------*/
void	HandleNull(void)
{
	int			numGPSPoints;
	int			numDataPoints;
	int			GPSbuff;
	long			numCharOut, numCharIn, numCharInOld;
	char			addressBuf[bufferSize];
	char			*tok;
	char			*timeOfRx;
	char			*latitude;
	char			*longitude;
	char			*HDOP;
	char			*delim=" ,";
	char			*end;
```

```
	int			i;
	unsigned long int	timeVal;
	double			latVal,longVal,HDOPVal;
	double			cosfact;
	unsigned long		t_mu;
	long			maxtime;
	Str255			numstr;
	char			*lattest="1234.89";
	double			lattestd, d1;
	long			lattestdec,lattestr;
	double			timemod;

	if (gDataScan == true)
	{

/*---> use this part when the GPS receiver is available

		numCharOut=5L;
		FSWrite(gASerRefOut, &numCharOut, "#GPS ");
		SerGetBuf(gASerRefIn,&numCharIn);
		maxtime=TickCount()+20L;
		while(numCharIn < 9L)
		{
			SerGetBuf(gASerRefIn,&numCharIn);
			if(TickCount() > maxtime) break;
		}
		FSRead(gASerRefIn,&numCharIn,&addressBuf);

		numCharOut=2L;
		FSWrite(gASerRefOut, &numCharOut, "?L");
		SerGetBuf(gASerRefIn,&numCharIn);
		maxtime=TickCount()+20L;
		while(numCharIn < 55L)
		{
			SerGetBuf(gASerRefIn,&numCharIn);
			if(TickCount() > maxtime) break;
		}
		FSRead(gASerRefIn,&numCharIn,&gGPSBuffer);
*/

		numCharOut=5L;
		FSWrite(gASerRefOut, &numCharOut, "#PTR ");
		SerGetBuf(gASerRefIn,&numCharIn);
		maxtime=TickCount()+20L;
		while(numCharIn < 9L)
		{
			SerGetBuf(gASerRefIn,&numCharIn);
			if(TickCount() > maxtime) break;
		}
		FSRead(gASerRefIn,&numCharIn,&addressBuf);

		numDataPoints=TalkSail('-');
```

```
		if(numDataPoints > 5)
		{
			gPulseCounter++;

			tok=strtok(gGPSBuffer,delim);
			timeOfRx=strtok(NULL,delim);
			latitude=strtok(NULL,delim);
			tok=strtok(NULL,delim);
			longitude=strtok(NULL,delim);
			tok=strtok(NULL,delim);
			tok=strtok(NULL,delim);
			tok=strtok(NULL,delim);
			HDOP=strtok(NULL,delim);

			timeVal=strtoul(timeOfRx,&end,10);
			latVal=strtod(latitude,&end);
			longVal=strtod(longitude,&end);
			HDOPVal=strtod(HDOP,&end);

			if (HDOPVal < 10.0)
			{
				if (gstarter == true)
				{
					t_mu=HandleDataIn(numDataPoints);
					gtime2=DecTimeToSec(timeVal);
					gX2=0.0;
					gY2=0.0;
					if (greflag == true)
					{
						gYref=LatToY(latVal);
						gYFact=cos((gYref/(1853.2*60.0))*kpi/180.0);
						gXref=LongToX(longVal,gYFact);
						greflag=false;
					}
					gstarter=false;
				}
				else
				{
					t_mu=HandleDataIn(numDataPoints);
/*---> use this if GPS availble to see if fixes are 2 sec apart

					if(DecTimeToSec(timeVal) != gtime2-2UL)
*/
					if(DecTimeToSec(timeVal) != gtime2)

					{
						gstarter=true;
					}
					else
					{
						gtime1=gtime2;
						gtime2=DecTimeToSec(timeVal);
```

```
                    gX1=gX2;
                    gX2=LongToX(longVal,gYFact)-gXref;
                    gY1=gY2;
                    gY2=LatToY(latVal)-gYref;

//---> make up data to test program

                    gw[gi]=gxold+cos(0.00515*gj)*5.14778;
                    gw[gi+1]=gyold+sin(0.00515*gj)*5.14778;

                    gxold=gw[gi];
                    gyold=gw[gi+1];
                    gj=gj+1.0;
                    d1=sqrt((gxold-1000)*(gxold-1000)
                            +(gyold-2000)*(gyold-2000)+3000*3000);
                    timemod=d1/1.5;
                    while(timemod > 2000.0)
                            timemod=timemod-2000.0;
                    gw[gi+2]=timemod;

//---> end of simulated data

/*--->use this if GPS available

                    gw[gi]=(1.0-t_mu/((gtime2-gtime1)*1000.0)*gX1
                            +gX2;
                    gw[gi+1]=(1.0-t_mu/((gtime2-gtime1)*1000.0)*gY1
                            +gY2;
                    gw[gi+2]=t_mu;
*/

                    gi=gi+3L;
                    if(gi >= 3L*(kStep*kNumExp+1L) )
                            trackerStart=true;
                    if(gi >= 3L*kStep*(kNumExp+1L) )
                    {
                            gi=0L;
                            gDataWrap=true;
                    }

                    if((trackerStart == true) || (gDataWrap == true))
                    {
                            tracker(gnumExp);

                    }

                }
            }
        }
        else
        {
            t_mu=HandleDataIn(numDataPoints);
```

```
                        gstarter=true;
                    }
                }
        }
}
/*-----------------------------------------------------------------*/


/*----------------------DecTimeToSec--------------------------------*/
unsigned long    DecTimeToSec(unsigned long timeVal)
{
        unsigned long    hr, min, sec, timeSec;

        hr=timeVal/10000UL;
        min=(timeVal-hr*10000UL)/100UL;
        sec=timeVal-hr*10000UL-min*100UL;

        timeSec=hr*3600UL+min*60UL+sec;

        return(timeSec);
}

/*-----------------------------------------------------------------*/


/*----------------------LongToX------------------------------------*/
double           LongToX( double longVal, double cosfactor)
{
        long double      deg, min, X;

        deg=floor(longVal/100.0);
        min=(longVal-deg*100.0);

        X=(deg*60.0+min)*1853.2*cosfactor;

        return(X);
}

/*-----------------------------------------------------------------*/


/*----------------------LatToY-------------------------------------*/
double           LatToY(double latVal)
{
        long double      deg, min, Y;

        deg=floor(latVal/100.0);
        min=(latVal-deg*100.0);

        Y=(deg*60.0+min)*1853.2;

        return(Y);
```

```
}

/*------------------------------------------------------------------*/


/*---------------------HandleMouseDown-----------------------------*/
void    HandleMouseDown(EventRecord *eventPtr)
{
        WindowPtr       whichWindow;
        GrafPtr         oldPort;
        short           thePart;
        long            menuChoice;

        thePart = FindWindow(eventPtr->where, &whichWindow);

        switch(thePart)
        {
                case inMenuBar:
                        menuChoice = MenuSelect( eventPtr->where );
                        HandleMenuChoice( menuChoice );
                        break;
                case inSysWindow:
                        SystemClick(eventPtr, whichWindow);
                        break;
                case inContent:
                        SetPort(whichWindow);
                        GlobalToLocal( &eventPtr->where);
                        break;
                case inDrag:
                        DragWindow(whichWindow, eventPtr->where,
                                                    &screenBits.bounds);
                        break;
                case inGoAway:
                        gDone = true;
                        break;
        }
}
/*------------------------------------------------------------------*/


/*----------------------HandleMouseUp------------------------------*/
void    HandleMouseUp(EventRecord *eventPtr)
{
        // Put some function in here
}
/*------------------------------------------------------------------*/


/*------------------HandleMenuChoice-------------------------------*/
void    HandleMenuChoice(long menuChoice)
{
        short           menu;
```

```
	short			item;

	if (menuChoice != 0)
	{
		menu=HiWord(menuChoice);
		item=LoWord(menuChoice);

		switch(menu)
		{
			case mApple:
				HandleAppleChoice(item);
				break;
			case mFile:
				HandleFileChoice(item);
				break;
			case mAttenuator:
				HandleAttenuator(item);
				break;
			case mBaud:
				HandleBaud(item);
				break;
			case mSignal:
				HandleSignal(item);
				break;
		}

		HiliteMenu(0);
	}
}
/*-------------------------------------------------------------------*/


/*----------------------HandleAppleChoice-----------------------------*/
void	HandleAppleChoice( short item )
{
	MenuHandle		appleMenu;
	Str255			accName;
	short			accNumber;

	switch(item)
	{
		case iAbout:
			NoteAlert(kBaseResID,nil);
			break;
		default:
			appleMenu = GetMHandle(mApple);
			GetItem(appleMenu, item,accName);
			accNumber=OpenDeskAcc(accName);
			break;
	}
}
/*-------------------------------------------------------------------*/
```

```
/*-------------------HandleFileChoice----------------------------*/
void    HandleFileChoice(short item)
{
        switch(item)
        {
                case iQuit:
                        gDone = true;
                        break;
        }
}
/*-------------------------------------------------------------------*/


/*--------------------------HandleAttenuator----------------------------*/
void    HandleAttenuator(short item)
{
        MenuHandle      menuHandle;
        long            numCharOut, numCharIn;
        char            attenBuff[10], addressBuf[10];

        numCharOut=5L;
        FSWrite(gASerRefOut, &numCharOut, "#PTR ");
        SerGetBuf(gASerRefIn,&numCharIn);
        while(numCharIn < 9L)
        {
                SerGetBuf(gASerRefIn,&numCharIn);
        }
        FSRead(gASerRefIn,&numCharIn,&addressBuf);

        numCharOut=2L;
        numCharIn=6L;
        menuHandle=GetMHandle(mAttenuator);

        CheckItem(menuHandle,gLastAttn,false);
        CheckItem(menuHandle,item,true);
        gLastAttn=item;

        switch(item)
        {
                case iAuto:
                        FSWrite(gASerRefOut, &numCharOut, "AA");
                        break;
                case i0dB:
                        FSWrite(gASerRefOut, &numCharOut, "A0");
                        break;
                case i3dB:
                        FSWrite(gASerRefOut, &numCharOut, "A1");
                        break;
                case i9dB:
                        FSWrite(gASerRefOut, &numCharOut, "A2");
                        break;
```

```
		case i12dB:
			FSWrite(gASerRefOut, &numCharOut, "A3");
			break;
		case i18dB:
			FSWrite(gASerRefOut, &numCharOut, "A4");
			break;
		case i24dB:
			FSWrite(gASerRefOut, &numCharOut, "A5");
			break;
		case i30dB:
			FSWrite(gASerRefOut, &numCharOut, "A6");
			break;
	}
	FSRead(gASerRefIn, &numCharIn, attenBuff);
}
/*---------------------------------------------------------------------*/


/*-------------------------HandleBaud---------------------------------*/
void	HandleBaud(short item)
{
	MenuHandle	menuHandle;
	long		numCharOut, numCharOutr, numCharIn;
	int		serConfig;
	OSErr		resetErrIn, resetErrOut;
	char		inBuff[8];
	SerStaRec	serialStatusOut;

	numCharOut=8L;
	menuHandle=GetMHandle(mBaud);

	CheckItem(menuHandle,gLastBaud,false);
	CheckItem(menuHandle,item,true);
	gLastBaud=item;

	switch(item)
	{
		case i300:
			FSWrite(gASerRefOut, &numCharOut, "#0000300");
			gserConfig=baud300+data8+stop10+noParity;
			break;
		case i600:
			FSWrite(gASerRefOut, &numCharOut, "#0000600");
			gserConfig=baud600+data8+stop10+noParity;
			break;
		case i1200:
			FSWrite(gASerRefOut, &numCharOut, "#0001200");
			gserConfig=baud1200+data8+stop10+noParity;
			break;
		case i2400:
			FSWrite(gASerRefOut, &numCharOut, "#0002400");
			gserConfig=baud2400+data8+stop10+noParity;
```

```
				break;
			case i4800:
				FSWrite(gASerRefOut, &numCharOut, "#0004800");
				gserConfig=baud4800+data8+stop10+noParity;
				break;
			case i9600:
				FSWrite(gASerRefOut, &numCharOut, "#0009600");
				gserConfig=baud9600+data8+stop10+noParity;
				break;
	}

	SerStatus(gASerRefOut,&serialStatusOut);
	while ((serialStatusOut.rdPend != 0) || (serialStatusOut.wrPend !=0))
	{
		SerStatus(gASerRefOut,&serialStatusOut);
	}

	FSRead(gASerRefIn, &numCharOut, &inBuff);

	resetErrOut=SerReset(gASerRefOut,gserConfig);
	resetErrIn=SerReset(gASerRefIn,gserConfig);			// Set the protocal
		if ((resetErrIn != noErr) || (resetErrOut != noErr))
		{
			SysBeep(10);
			ScrollDisplay();
			DrawString("\pProblem setting the serial protocal.");
		}
}
/*----------------------------------------------------------------------*/


/*------------------------HandleSignal----------------------------------*/
void	HandleSignal(short item)
{
	Str255		periodData="\p!M52 ";
	Str255		numMaxData="\p!M63 ";
	Str255		windowData="\p!M74 ";
	char		hexChar;
	long		numCharOut, numCharIn;
	char		inbuff[30], outbuff[20], addressBuf[10];
	int		cr[1] ={13};
	int		radix, hexDec;
	short		i;
	int		period, numMax, window;
	SerStaRec	serialStatusOut;

	numCharOut=5L;
	FSWrite(gASerRefOut, &numCharOut, "#PTR ");
	SerGetBuf(gASerRefIn,&numCharIn);
	while(numCharIn < 9L)
	{
		SerGetBuf(gASerRefIn,&numCharIn);
```

```
}
FSRead(gASerRefIn,&numCharIn,&addressBuf);

switch(item)
{
        case iPeriod:
                gPeriod=HandleDialog(kPeriodDialog);
                period=gPeriod;
                for (i=0;i < k16Bit; i++)
                {
                        radix=Power(kHexBase,k16Bit-1-i);
                        hexDec=period/radix;
                        hexChar=DecIntToHexChar(hexDec);
                        periodData[1+i+periodData[0]]=hexChar;
                        period=period-hexDec*radix;
                }
                periodData[periodData[0]+k16Bit+1]=cr[0];
                periodData[0]=periodData[0]+k16Bit+1;

                for (i=0;i<periodData[0];i++)
                {
                        outbuff[i]=periodData[i+1];
                }
                numCharOut=periodData[0];
                break;
        case iNumMax:
                gNumMax=HandleDialog(kNumMaxDialog);
                numMax=gNumMax*2;
                for (i=0;i < k8Bit; i++)
                {
                        radix=Power(kHexBase,k8Bit-1-i);
                        hexDec=numMax/radix;
                        hexChar=DecIntToHexChar(hexDec);
                        numMaxData[1+i+numMaxData[0]]=hexChar;
                        numMax=numMax-hexDec*radix;
                }
                numMaxData[numMaxData[0]+k8Bit+1]=cr[0];
                numMaxData[0]=numMaxData[0]+k8Bit+1;

                for (i=0;i<numMaxData[0];i++)
                {
                        outbuff[i]=numMaxData[i+1];
                }
                numCharOut=numMaxData[0];
                break;
        case iWindow:
                gWindow=HandleDialog(kWindowDialog);
                window=gWindow;
                for (i=0;i < k16Bit; i++)
                {
                        radix=Power(kHexBase,k16Bit-1-i);
                        hexDec=window/radix;
```

```
					hexChar=DecIntToHexChar(hexDec);
					windowData[1+i+numMaxData[0]]=hexChar;
					window=window-hexDec*radix;
				}
				windowData[windowData[0]+k16Bit+1]=cr[0];
				windowData[0]=windowData[0]+k16Bit+1;

				for (i=0;i<windowData[0];i++)
				{
					outbuff[i]=windowData[i+1];
				}
				numCharOut=windowData[0];
				break;
	}
	FSWrite(gASerRefOut,&numCharOut,&outbuff);

	while(1 == 1)
	{
		SerGetBuf(gASerRefIn,&numCharIn);
		FSRead(gASerRefIn, &numCharIn, &inbuff);

		if(inbuff[numCharIn-1L] == 03) break;
	}


}
/*-----------------------------------------------------------------*/


/*-------------------DecIntToHexChar-----------------------------------*/
int			DecIntToHexChar(int decimal)
{
	int		hexChar;

	if ((decimal >= 0) & (decimal < 10))
		hexChar=48+decimal;
	if ((decimal >= 10) & (decimal < 16))
		hexChar=55+decimal;

	return(hexChar);
}
/*-----------------------------------------------------------------*/


/*-------------------ConcatString--------------------------------------*/
void	ConcatString(Str255 str1, Str255 str2)
{
	short	i;

	for(i=str1[0];i<str2[0]+str1[0];i++)
	{
		str1[i+1]=str2[i-str1[0]+1];
```

```
	}
	str1[0]=i;
}
/*------------------------------------------------------------------*/



/*--------------------HandleDialog----------------------------------*/
unsigned long		HandleDialog(int dialogRes)
{
	WindowPtr	window;
	DialogPtr	dialog;
	Boolean		dialogDone;
	Handle		textItemHandle;
	Handle		okItemHandle;
	short		itemType, itemHit;
	Rect		itemRect;
	long		itemNum;
	unsigned long	itemNumU;

	window=FrontWindow();
	dialog=GetNewDialog(dialogRes,nil,kMoveToFront);
	ShowWindow(dialog);
	SetPort(dialog);
/*	SetDialogDefaultItem(dialog, ok);
	SetDialogCancelItem(dialog, cancel);
	SetDialogTracksCursor(dialog, true);
*/

	GetDItem(dialog, iMessageText, &itemType,
				&textItemHandle, &itemRect);
	GetDItem(dialog, ok, &itemType, &okItemHandle,
				&itemRect);
	SelIText(dialog, iMessageText, kMinTextPosition, kMaxTextPosition);

	dialogDone=false;
	while( ! dialogDone)
	{
		GetIText(textItemHandle, &gItemText);
		ModalDialog(nil, &itemHit);

		switch(itemHit)
		{
			case ok:
			case cancel:
				dialogDone=true;
				break;
		}
	}

	DisposeDialog(dialog);
```

```
		SelectWindow(window);
		SetPort(window);

		StringToNum(gItemText, &itemNum);
		itemNumU=itemNum;
		return(itemNumU);

}
/*-----------------------------------------------------------------*/


/*-------------------HandleKeyIn-----------------------------------*/
void	HandleKeyIn(char theChar)
{
		int			bufferLength, i;
		RgnHandle	gtempRgn;
		long		numCharOut, numCharIn;
		char		addressBuf[10];

		gtempRgn = NewRgn();

		switch(theChar)
		{
				case 'W':
						gDataScan=false;
						SelectWindow(gTerminalWind);
						SetPort(gTerminalWind);
						DrawString("\pW");
						ScrollDisplay();
						DrawString("\p:");
						break;
				case 'K':
						gDataScan=false;
						SelectWindow(gTerminalWind);
						SetPort(gTerminalWind);
						numCharOut=5L;
						FSWrite(gASerRefOut, &numCharOut, "#PTR ");
						SerGetBuf(gASerRefIn,&numCharIn);
						while(numCharIn < 9L)
						{
								SerGetBuf(gASerRefIn,&numCharIn);
						}
						FSRead(gASerRefIn,&numCharIn,&addressBuf);

						bufferLength=TalkSail('P');
						DisplayString(bufferLength);
						gi=0L;
						break;
				case '-':
						gDataScan=true;
						SelectWindow(gWetPaperWind);
						SetPort(gWetPaperWind);
```

```
				for (i=0;i < 10;i++)
				{
					ScrollDataScan();
				}
				numCharOut=5L;
				FSWrite(gASerRefOut, &numCharOut, "#PTR ");
				SerGetBuf(gASerRefIn,&numCharIn);
				while(numCharIn < 9L)
				{
					SerGetBuf(gASerRefIn,&numCharIn);
				}
				FSRead(gASerRefIn,&numCharIn,&addressBuf);

				bufferLength=TalkSail('O');
				bufferLength=TalkSail(theChar);
				break;
			default:
				SelectWindow(gTerminalWind);
				SetPort(gTerminalWind);
				bufferLength=TalkSail(theChar);
				DisplayString(bufferLength);
				break;
	}

}
/*--------------------------------------------------------------------*/


/*--------------------HandleDataIn------------------------------------*/
unsigned long		HandleDataIn(int numDataPoints)
{
	int			i, j;
	char			max[8][4], time[8][4];
	unsigned		timeInt[8], maxInt[8];
	unsigned long	timeLong;
	Str255		timeOfMaxString;
	unsigned long	drawTime, dtemp, horizontalPosit;
	float			fraction;
	Str255		numpixel,horstring;

	for (j=0;j<gNumMax;j++)
	{
		for (i=0;i<4;i++)
		{
			max[j][i]=gDataBuffer[8*j+4-i];
			time[j][i]=gDataBuffer[8*j+8-i];
		}

		timeInt[j]=0;
		maxInt[j]=0;
		for(i=0;i<4;i++)
		{
```

```
				timeInt[j]=timeInt[j]+HexCharToInt(time[j][i],i);
				maxInt[j]=maxInt[j]+HexCharToInt(max[j][i],i);
			}
		}

		for (i=0;i < gNumMax; i++)
		{
			drawTime=(timeInt[i]*(ghorizontalPixels-8UL))/gPeriod;

			SetPort(gWetPaperWind);
			horizontalPosit=drawTime+kLeftMargin;
			PenNormal();
			MoveTo(horizontalPosit,gDataRowStart);
			Line(0,0);
		}

		ScrollDataScan();

		timeLong=timeInt[0];
		return(timeLong);
}
/*-------------------------------------------------------------------*/


/*------------------HexCharToInt-------------------------------------*/
unsigned		HexCharToInt(char Bite,int exponent)
{
		unsigned		value;


		if ((Bite > 47) & (Bite < 58))
				value=(Bite-48)*Power(kHexBase,exponent);
		if ((Bite > 64) & (Bite < 71))
				value=(Bite-55)*Power(kHexBase,exponent);

		return(value);
}
/*-------------------------------------------------------------------*/


/*-----------------TalkSail-----------------------------------------*/
int			TalkSail(char theChar)
{
		long			numCharOut, numCharIn, startTime,maxTime;
		OSErr			serOutErr, serInErr;
		char			inBuff[bufferSize],DataBuffer[bufferSize],inChar;
		int			i, start;
		SerStaRec		serialStatusIn, serialStatusOut;
		Str255			startString;

		numCharOut=1L;
```

```
	serOutErr=FSWrite(gASerRefOut, &numCharOut, &theChar);
		if (serOutErr != noErr)
		{
			SysBeep(10);
			DrawString("\pProblem sending the character out.");
			ScrollDisplay();
			DrawString("\p:");
		}

	SerStatus(gASerRefOut,&serialStatusOut);
	while ((serialStatusOut.rdPend != 0) || (serialStatusOut.wrPend !=0))
	{
		SerStatus(gASerRefOut,&serialStatusOut);
	}

	SerStatus(gASerRefIn,&serialStatusIn);
	while(serialStatusIn.rdPend != 0)
			SerStatus(gASerRefIn,&serialStatusIn);

	start=0;
	startTime=TickCount();
	maxTime=startTime+20L;
	while(1 == 1)
	{
		SerGetBuf(gASerRefIn,&numCharIn);
		serInErr=FSRead(gASerRefIn, &numCharIn, &inBuff);
			if (serInErr != noErr)
			{
				SysBeep(10);
				DrawString("\pProblem receiving characters.");
				ScrollDisplay();
				DrawString("\p:");
			}

		memmove(&gDataBuffer[start],&inBuff,numCharIn);
		start=start+LoWord(numCharIn);

		if(inBuff[numCharIn-1L] == 03) break;
		if((start < 2) & (TickCount() > maxTime)) break;
	}
	return(start);
}
/*-------------------------------------------------------------------*/


/*--------------------DisplayString-----------------------------------*/
void		DisplayString(int numDataPoints)
{
	int	i;

	for (i=0;i < numDataPoints ;i++)
	{
```

```
			switch(gDataBuffer[i])
			{
				case 03:
					break;
				case 13:
					{
						if (gDataBuffer[i+1] == 10)
						{
							ScrollDisplay();
							i=i+1;
						}
						break;
					}
				case 10:
					{
						if (gDataBuffer[i+1] == 13)
						{
							ScrollDisplay();
							i=i+1;
						}
						break;
					}
				default:
					DrawChar(gDataBuffer[i]);
			}
		}
}
/*-----------------------------------------------------------------*/


/*--------------------ScrollDisplay--------------------------------*/
void	ScrollDisplay(void)
{
	RgnHandle	tempRgn;
	WindowPtr	window;

	window = FrontWindow();
	tempRgn = NewRgn();

	ScrollRect(&window->portRect, kHorizontalOffset,
					-kRowHeight, tempRgn);
	DisposeRgn(tempRgn);

	MoveTo(kLeftMargin, gRowStart);
}
/*-----------------------------------------------------------------*/


/*--------------------ScrollDataScan-------------------------------*/
void	ScrollDataScan(void)
{
	RgnHandle	tempRgn;
```

```
	WindowPtr	window;

	window = FrontWindow();

	tempRgn = NewRgn();

	ScrollRect(&window->portRect, kHorizontalOffset,
				-kDataScanHeight, tempRgn);
	DisposeRgn(tempRgn);

	MoveTo(kLeftMargin, gDataRowStart);
}
/*----------------------------------------------------------------*/


/*----------------------DoUpdate--------------------------------*/
void	DoUpdate(EventRecord *eventPtr)
{
	WindowPtr		window;

	window = (WindowPtr)eventPtr->message;

	BeginUpdate(window);
	// put some update info in here
	EndUpdate(window);
}
/*----------------------------------------------------------------*/


/*---------------------Power-------------------------------------*/
int		Power(int base, int exponent)
{
	int		i, p;

	p=1;
	for (i=1;i <= exponent;++i)
		p=p*base;
	return(p);
}
/*----------------------------------------------------------------*/


void		tracker(long	numExp)
{
	int		i, j;
	int		nextIteration, noUpdate=0, numIterations=0;
	long		u, index1, index2;
	double		f, Cs=1.5;
	double		theta[kParameters]={0.0,1000.0,-2500};
	double		xvar, yvar, tvar, V[kNumMeas];
	double		term;
	double		stopval=1.0, epsilon=0.1, epsilonc;
```

```
double          D[kParameters][kParameters];
double          Dinv[kParameters][kParameters];
double          dtheta[kParameters]={0.0,0.0,0.0};
double          dthetaSum[kParameters]={0.0,0.0,0.0};
double          Z0=0.0, Z1;
double          d1_x, d1_y, d_z, d1;
double          d2_x, d2_y, d2;
double          g[kNumExp+1], g1[kNumExp+1];
double          dthetaVec;
double          x1, x2, y1, y2;
double          lamda1, lamda2;
double          feas1, feas2, feas3, rhomax;
Boolean         nextIter=true;
double          *e, *a, *b, *c, *lamda;
double          *wtrue;
double          *dwtrue;
double          *wmeas;
Str255          numstr, thetaString;
long            thestring;
double          maxR=5000.0, maxD=-5000.0, minD=-1.0;

wtrue = calloc(3L*(kStep*kNumExp+1),sizeof(double));
if(wtrue == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}

wmeas = calloc(3L*(kStep*kNumExp+1),sizeof(double));
if(wmeas == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}

dwtrue = calloc(3L*(kStep*kNumExp+1),sizeof(double));
if(dwtrue == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}

e = calloc(kNumExp*kNumMeas,sizeof(double));
if (e == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}
a = calloc(kNumExp*kNumMeas,sizeof(double));
if (a == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
```

```
        abort();
}
b = calloc(kNumExp*kParameters,sizeof(double));
if (b == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}
c = calloc(kNumExp,sizeof(double));
if (c == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}
lamda = calloc(kNumExp,sizeof(double));
if (lamda == NULL)
{
        DrawString("\pCan't allocate memory for data collection!!!");
        abort();
}

f=1.0/sqrt(2.0);

for(u=0;u<kNumExp;u++)
{
        index1=3*(u*kStep+numExp);
        if(index1 >= 3*kStep*(kNumExp+1))
                index1=index1-3*kStep*(kNumExp+1);

        index2=3*((u+kNumExp/2)*kStep+numExp);
        if(index2 >= 3*kStep*(kNumExp+1))
                index2=index2-3*kStep*(kNumExp+1);

        for(i=0;i<3;i++)
        {
                wtrue[kNumMeas*u+i]=gw[index1+i];
                wtrue[kNumMeas*u+i+3]=gw[index2+i];
                wmeas[kNumMeas*u+i]=gw[index1+i];
                wmeas[kNumMeas*u+i+3]=gw[index2+i];
        }
}

while(stopval > epsilon)
{
        numIterations++;
        for(i=0;i<kParameters;i++)
        {
                dthetaSum[i]=0.0;
                for (j=0;j<3;j++)
                        D[i][j]=0.0;
        }
        Z0=0.0;
```

```
for(u=0;u<kNumExp;u++)
{
	for(i=0;i<kNumMeas;i++)
	{
		e[6*u+i]=wtrue[6*u+i]-wmeas[6*u+i];
	}

	d1_x=(wtrue[6*u]-theta[0])*(wtrue[6*u]-theta[0]);
	d1_y=(wtrue[6*u+1]-theta[1])*(wtrue[6*u+1]-theta[1]);
	d_z=theta[2]*theta[2];
	d1=sqrt(d1_x+d1_y+d_z);

	d2_x=(wtrue[6*u+3]-theta[0])*(wtrue[6*u+3]-theta[0]);
	d2_y=(wtrue[6*u+4]-theta[1])*(wtrue[6*u+4]-theta[1]);
	d2=sqrt(d2_x+d2_y+d_z);

	g[u]=d1-d2-Cs*wtrue[6*u+2]+Cs*wtrue[6*u+5];

	a[6*u+0]= (wtrue[6*u+0]-theta[0])/d1;
	a[6*u+1]= (wtrue[6*u+1]-theta[1])/d1;
	a[6*u+2]= -Cs;
	a[6*u+3]= -(wtrue[6*u+3]-theta[0])/d2;
	a[6*u+4]= -(wtrue[6*u+4]-theta[1])/d2;
	a[6*u+5]= Cs;


	b[3*u+0]=(wtrue[6*u+3]-theta[0])/d2-(wtrue[6*u+0]-theta[0])/d1;
	b[3*u+1]=(wtrue[6*u+4]-theta[1])/d2-(wtrue[6*u+1]-theta[1])/d1;
	b[3*u+2]=theta[2]/d1-theta[2]/d2;
	for(i=0;i<3;i++)
		if(b[3*u+i] == 0.0)
			b[3*u+i]=1e-6;

	c[u]=0.0;
	for(j=0;j<6;j++)
		term=a[6*u+j]*gV[j]*a[6*u+j];
	c[u]=c[u]+term;

	for(i=0;i<3;i++)
	{
		for(j=0;j<3;j++)
		{
			D[i][j]=D[i][j]+(1/c[u])*(b[3*u+i]*b[3*u+j]);
		}
	}

	dthetaVec=0.0;
	for(i=0;i<6;i++)
	{
		dthetaVec=dthetaVec+a[6*u+i]*e[6*u+i];
	}
```

```
		for(i=0;i<3;i++)
		{
			dthetaSum[i]=dthetaSum[i]+(dthetaVec-g[u])*b[3*u+i]/c[u];
		}

		Z0=Z0+g[u]*g[u]/c[u];
	}

	x1=D[1][1]-D[0][1]*D[1][0]/D[0][0];
	x2=D[1][2]-D[0][2]*D[1][0]/D[0][0];
	y1=D[2][1]-D[0][1]*D[2][0]/D[0][0];
	y2=D[2][2]-D[0][2]*D[2][0]/D[0][0];

	Dinv[2][2]=x1/(x1*y2-x2*y1);
	Dinv[1][2]=-x2/(x1*y2-x2*y1);
	Dinv[0][2]=-(D[0][1]*Dinv[1][2]+D[0][2]*Dinv[2][2])/D[0][0];

	Dinv[2][1]=-y1/(x1*y2-x2*y1);
	Dinv[1][1]=(1.0+x2*y1/(x1*y2-x2*y1))/x1;
	Dinv[0][1]=-(D[0][1]*Dinv[1][1]+D[0][2]*Dinv[2][1])/D[0][0];

	Dinv[2][0]=(x1/(x1*y2-x2*y1))*
			(-D[2][0]/D[0][0]+(D[1][0]*y1)/(D[0][0]*x1));
	Dinv[1][0]=-(D[1][0]/D[0][0]+x2*Dinv[2][0])/x1;
	Dinv[0][0]=(1.0-(D[0][1]*Dinv[1][0]+D[0][2]*Dinv[2][0]))/D[0][0];

	for(i=0;i<kParameters;i++)
	{
		dtheta[i]=0.0;
		for(j=0;j<kParameters;j++)
		{
			dtheta[i]=dtheta[i]+Dinv[i][j]*dthetaSum[j];
		}
	}

	for(u=0;u<kNumExp;u++)
	{
		lamda1=0.0;
		for(i=0;i<3;i++)
		{
			lamda1=lamda1+b[3*u+i]*dtheta[i];
		}
		lamda2=0.0;
		for(i=0;i<6;i++)
		{
			lamda2=lamda2+a[6*u+i]*e[6*u+i];
		}
		lamda[u]=(lamda1-lamda2+g[u])/c[u];

		for(i=0;i<6;i++)
		{
//			term=gV[i]*a[6*u+i];
```

```
                dwtrue[6*u+i]=-e[6*u+i]-lamda[u]*gV[i]*a[6*u+i];
        }
}

if(dtheta[0] >= 0.0)
        feas1=(kMaxR-theta[0])/dtheta[0];
if(dtheta[0] < 0.0)
        feas1=(kMaxR+theta[0])/(-dtheta[0]);

if(dtheta[1] >= 0.0)
        feas2=(kMaxR-theta[1])/dtheta[1];
if(dtheta[1] < 0.0)
        feas2=(kMaxR+theta[1])/(-dtheta[1]);

if(dtheta[2] >= 0.0)
        feas3=(kMinD-theta[2])/dtheta[2];
if(dtheta[2] < 0.0)
        feas3=(kMaxD-theta[2])/dtheta[2];

rhomax=1.0;
if(feas1 < rhomax)
        rhomax = feas1;
if(feas2 < rhomax)
        rhomax = feas2;
if(feas3 < rhomax)
        rhomax=feas3;

epsilon=(gtheta[0]+0.001)/1e6;
if(epsilon < 0)
        epsilon=-epsilon;
epsilonc=(gtheta[1]+0.001)/1e6;
if(epsilonc < 0)
        epsilonc=-epsilonc;
if(epsilonc <  epsilon)
        epsilon=epsilonc;
epsilonc=(gtheta[2]+0.001)/1e6;
if(epsilonc < 0)
        epsilonc=-epsilonc;
if(epsilonc <  epsilon)
        epsilon=epsilonc;

nextIteration=0;
while(nextIteration == 0)
{
        Z1=0.0;
        for(u=0;u<kNumExp;u++)
        {
                d1_x=((wtrue[6*u]+rhomax*f*dwtrue[6*u])
                                -(gtheta[0]+rhomax*f*dtheta[0]))*
                        ((wtrue[6*u]+rhomax*f*dwtrue[6*u])
                                -(gtheta[0]+rhomax*f*dtheta[0]));
                d1_y=((wtrue[6*u+1]+rhomax*f*dwtrue[6*u+1])
```

```
                    -(gtheta[1]+rhomax*f*dtheta[1]))*
                ((wtrue[6*u+1]+rhomax*f*dwtrue[6*u+1])
                    -(gtheta[1]+rhomax*f*dtheta[1]));
            d_z=(gtheta[2]+rhomax*f*dtheta[2])*(gtheta[2]+rhomax*f*dtheta[2]);
            d1=sqrt(d1_x+d1_y+d_z);

            d2_x=((wtrue[6*u+3]+rhomax*f*dwtrue[6*u+3])
                    -(gtheta[0]+rhomax*f*dtheta[0]))*
                ((wtrue[6*u+3]+rhomax*f*dwtrue[6*u+3])
                    -(gtheta[0]+rhomax*f*dtheta[0]));
            d2_y=((wtrue[6*u+4]+rhomax*f*dwtrue[6*u+4])
                    -(gtheta[1]+rhomax*f*dtheta[1]))*
                ((wtrue[6*u+4]+rhomax*f*dwtrue[6*u+4])
                    -(gtheta[1]+rhomax*f*dtheta[1]));
            d2=sqrt(d2_x+d2_y+d_z);

            g[u]=d1-d2
                -Cs*(wtrue[6*u+2]+rhomax*f*dwtrue[6*u+2])
                +Cs*(wtrue[6*u+5]+rhomax*f*dwtrue[6*u+5]);

            Z1=Z1+g1[u]*g1[u]/c[u];
        }

        if(noUpdate > 5) Z1=0.0;
        if(Z1 < Z0)
        {
            noUpdate=0;
            nextIteration=1;
            for(i=0;i<3;i++)
            {
                theta[i]=theta[i]+rhomax*f*dtheta[i];
                gPosition[i]=gPosition[i]+theta[i];
            }
            for(u=0;u<kNumExp;u++)
            {
                for(i=0;i<6;i++)
                    wtrue[6*u+i]=wtrue[6*u+i]
                            +rhomax*f*dwtrue[6*u+i];
            }
        }
        else
            rhomax=rhomax/2.0;
            noUpdate++;
    }

    MoveTo(50,150);
    thestring=theta[0];
    NumToString(thestring,&numstr);
    DrawString(numstr);
    DrawString("\p  :  ");
    thestring=theta[1];
    NumToString(thestring,&numstr);
```

```
			DrawString(numstr);
			DrawString("\p  :  ");
			thestring=theta[2];
			NumToString(thestring,&numstr);
			DrawString(numstr);
			for(i=0;i<10;i++) ScrollDataScan();

			if(numIterations > 50)
				stopval=0.0;
			else
			{
				if(dtheta[0] < 0)
					stopval=-dtheta[0];
				else
					stopval=dtheta[0];
			}
		}

		if(numExp < (kNumExp+1)*kStep-1)
			gnumExp=gnumExp+1L;
		else
			gnumExp=0L;


		free(wtrue);
		free(wmeas);
		free(dwtrue);
		free(e);
		free(a);
		free(b);
		free(c);
		free(lamda);
	}
```

References

Allen, Mark R. and King, Louis A.. 1988. An adaptive Two Stage Kalman Structure for Passive Undersea Tracking. IEEE Transactions on Acoustics, Speech, and Signal Processing. Vol. 36, No 1, pages 3-9

Bard, Yonathan. 1974. Nonlinear Parameter Estimation. New York: Academic Press Inc.

Cheng, David K. 1983. Field and Wave Electromagnetic. Reading, Massachusetts: Addison-Wesley Publishing Company.

Friedlander, Benjamin. 1987. A Passive Localization Algorithm and Its Accuracy. IEEE Journal of Oceanic Engineering. Vol. OE-12, No. 1, pages 234-245

Horowitz and Hill. 1989. The Art of Electronics. New York: Cambridge University Press.

Millman, Jacob. 1979. Micro-Electronics: Digital and Analog Circuits and Systems. New York: McGraw-Hill Book Company.

Moose, Richard L.. 1987. Passive Range Estimation of an Underwater Maneuvering Target. IEEE Transactions on Acoustics, Speech, and Signal Processing. Vol. ASSP-35, No 3, pages 274-2859

Wenz, J.. 1962. Ambient Noise Spectra. Journal of the Acoustical Society of America. Vol. 34, p 1936.

Wilson, Oscar Bryan 1985. An Introduction To The Theory and Design of Sonar Transducers Monterey, California: Naval Postgraduate School.